Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Рубан
подпись
«_____» _____ 2016 г.

ДИПЛОМНЫЙ ПРОЕКТ

230105.65 Программное обеспечение вычислительной техники
и автоматизированных систем

Автоматизированная информационная система учёта заказов на предприятии

Пояснительная записка

Научный руководитель _____ ст. преподаватель А.В. Прокопенко
подпись, дата

Нормоконтролер _____ доцент, к. т. н. О. А. Антамошкин
подпись, дата

Выпускник _____ Ю.В. Липовка
подпись, дата

Красноярск 2016

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
Кафедра «Информатика»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А. И. Рубан
подпись
«_____» _____ 2016 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме дипломного проекта

Студенту Липовка Юрию Викторовичу

Группа ЗКИ10-05 Направление (специальность) 230105.65, Программное обеспечение вычислительной техники и автоматизированных систем.

Тема выпускной квалификационной работы: «Автоматизированная информационная система учёта заказов на предприятии».

Утверждена приказом по университету № 4202/с от 28.03.2016 г. Руководитель: Прокопенко А.В., к.т.н., ст. преподаватель кафедры "Информатика" ИКИТ СФУ

Исходные данные для ВКР: спроектировать и разработать автоматизированную информационную систему учета заказов на предприятии для ООО «ЭлектроПлюс»

Перечень разделов ВКР:

- введение;
- аналитическая часть;
- проектная часть;
- описание работы приложения;
- эргономика.

Перечень графического или иллюстративного материала с указанием основных чертежей, плакатов, слайдов: презентационные слайды PowerPoint.

Руководитель ВКР _____ А.В. Прокопенко
(подпись)

Задание принял к исполнению _____ Ю.В. Липовка
(подпись)

« ___ » _____ 2016 г.

АННОТАЦИЯ

Выпускная квалификационная работа по теме «Автоматизированная информационная система учёта заказов на предприятии» содержит 80 страниц текстового документа, включая 2 приложения, 38 рисунков, 14 библиографических источников.

Целью работы является разработка программного комплекса, предназначенного для ведения информационной системы учета и исполнения заказов предприятия ООО «ЭлектроПлюс».

Информационная система представляет собой клиент-серверное приложение, клиентская часть которого разработана на языке программирования C# в среде разработки Microsoft Visual Studio 2015 и взаимодействует с СУБД MySQL.

В дипломный проект входит введение, четыре основные части и заключение.

Во введении определяется необходимость реализации и основные задачи проекта.

В первой части проекта описывается постановка задачи, выбор языка программирования, описание основных бизнес-процессов и проектирование информационной системы.

Во второй части проекта описана проектная часть, разработка программных модулей, таблиц баз данных, отчетов системы.

В третьей части описано приложение, его основные формы.

В четвертой части затронута эргономика приложения.

Заключение посвящено подведению итогов по всей проделанной работе.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Ни одну область деятельности человека, поддерживаемую информационными технологиями, невозможно представить себе без использования баз данных и информационных систем, помогающих получить быстрый доступ к информации, увеличивая тем самым продуктивность работы.

Использование баз данных и информационных систем становится неотъемлемой составляющей деловой деятельности современного человека и функционирования преуспевающих организаций.

Разрабатываемая в данном дипломном проекте информационная система является важным решением для ООО «ЭлектроПлюс», и преследует цель повысить эффективность работы сотрудников и планирования экономической деятельности этого предприятия.

Основной целью построения автоматизированной информационной системы электромонтажной фирмы является хранение информации о деятельности фирмы (ее заказах), о ее сотрудниках и заказчиках.

Большинство задач предприятий являются рутинными, которые без проблем можно возложить на «плечи систем автоматизации». При этом повысив уровень эффективности работы персонала, сокращения временных и финансовых издержек, а самое главное иметь возможность сосредоточения на основных бизнес процессах предприятия.

Автоматизированная информационная система учета заказов предназначена для оптимизации работы, в первую очередь, руководства и управляющего персонала и играет большую роль в повышении производительности их труда.

Руководителям предприятия система поможет оперативно проводить анализ работы предприятия, проводить экономические и финансовые расчеты, проводить анализ данных по разным категориям заказов и услуг, что, несомненно, положительно скажется при принятии решений.

1 Аналитическая часть

1.1 Описание предметной области

Автоматизированная информационная система - это система, предназначенная для хранения, поиска и обработки необходимой информации. Она не может обойтись без соответствующих организационных ресурсов (человеческих, технических, финансовых и т.д.), которые обеспечивают и распространяют информацию. Таким образом, автоматизированная информационная система - это совокупность баз данных, систем управления базами данных и прикладных программ, функционирующих на вычислительных средствах как единое целое для решения определенных задач[1].

Данная система, описанная в этом проекте, предназначена для учета деятельности предприятия ООО «ЭлектроПлюс», хранения заказов об услугах фирмы, а также для хранения данных о заказчиках и сотрудниках.

Функциональные требования к программным модулям:

1. учет заказов фирмы;
2. хранение персональных данных о сотрудниках;
3. хранение персональных данных о заказчиках;
4. формирование списка предоставляемых фирмой услуг;
5. формирование данных по сотрудникам и исполняемых ими заказах;
6. формирование отчетов по заказам.

Процесс проектирования информационных систем является достаточно сложной задачей. Он начинается с построения инфологической модели данных, то есть идентификации сущностей. Затем необходимо выполнить следующие шаги процедуры проектирования даталогической модели, то есть мифологическая модель должна быть отображена в компьютер-ориентированную даталогическую модель, «понятную» СУБД[2].

Представить предметную область в виде совокупности отдельных независимых друг от друга объектов, каждый из которых будет описываться

своей таблицей. Для каждой таблицы определить ключевые поля; установить связи между таблицами; для каждой связи определить тип. Разработать структуру каждой таблицы: перечень полей, их типы и свойства. Заполнить таблицы данными. Разработать необходимые запросы к базе данных, входные и выходные формы и отчеты. Предусмотреть возможность автоматизации часто выполняемых действий путем создания макросов и программных модулей.

База данных фирмы по электромонтажным услугам должна содержать следующую информацию о сотрудниках:

Данные о сотруднике:

1. Фамилия, Имя, Отчество;
2. Серия паспорта;
3. Номер паспорта;
4. Рабочий телефон;
5. E-mail;
6. Должность;
7. Дата рождения;
8. Адрес проживания;

Целью деятельности фирмы является привлечение как можно большего количества заказчиков. В базе данных необходимо иметь следующую информацию о них:

Данные о заказчике:

1. Фамилия, Имя, Отчество;
2. Адрес проживания;
3. Рабочий телефон;
4. E-mail;

В ходе своей деятельности сотрудники фирмы выполняют работы на объектах, у которых есть наименование и адрес. Добавим в модель базы данных следующую информацию:

Данные об объекте:

1. Адрес объекта
2. Наименование (название) обьекта.

Фирма выполняет свои услуги, у каждой из которых есть название, и ее можно отнести к какой либо категории услуг. Составим следующий список:

Данные об услуге:

1. Наименование услуги;
2. Категория;
3. Единица измерения услуги;
4. Стоимость за единицу (руб.);

Фирма может покупать кабель, инструменты и материалы у разных поставщиков. Следовательно, необходим список товаров и их описание:

Данные о товаре:

1. Наименование товара;
2. Единица измерения;
3. Стоимость товара;
4. Наименование поставщика;
5. Адрес поставщика;
6. ИНН поставщика;
7. КПП поставщика;

Для учета поступления заказов, необходимо располагать всей нужной информацией, а именно наименование заказчика, исполнителя, номер заказа, даты исполнения и т.д.:

Данные о заказах фирмы:

1. Номер заказа;
2. Дата заказа;
3. Сроки выполнения заказа;

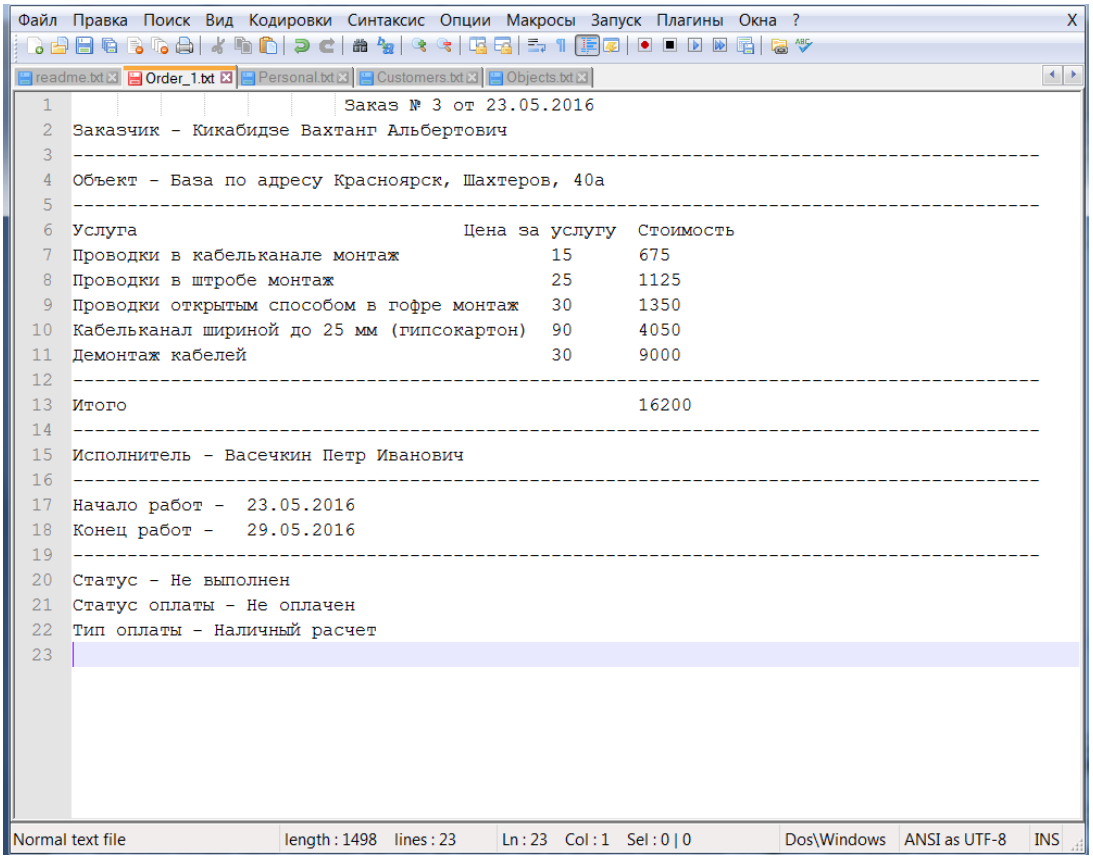
4. Дата начала работы;
5. Итоговая сумма заказа;
6. Предоплата;
7. Вид оплаты;
8. Проплачен заказ;
9. Заказчик;
10. Объект;
11. Услуги и их количество;

1.2 Постановка задачи проектирования

Определим список задач, которая будет решать информационная система:

- учет заказов о предоставлении услуг клиентам, управление доходами/расходами;
- автоматическая печать документов: счетов на оплату, сохранение в файл документов;
- просмотр заказов, с подробным отчетом по заказанным работам;
- формирование списка предоставляемых фирмой услуг;
- формирование данных по сотрудникам и исполняемых ими заказах;
- возможность поиска заказов по различным категориям;

В фирме «ЭлектроПлюс» клиентам представляются следующие документы: отчет о заказе в произвольной форме (txt файл), прайс-лист об услугах. В любой момент времени клиент должен получить печать документа о заказе (рис. 1). Администрация фирмы должна получить в печатном или электронном формате отчеты о заказах, списках сотрудников, объектов, товарах.




```
                    Заказ № 3 от 23.05.2016
Заказчик - Кикабидзе Вахтанг Альбертович
-------------------------------------------------------------------------------------------------
Объект - База по адресу Красноярск, Шахтеров, 40а
-------------------------------------------------------------------------------------------------
Услуга                                  Цена за услугу  Стоимость
Проводки в кабельканале монтаж                  15      675
Проводки в штробе монтаж                        25      1125
Проводки открытым способом в гофре монтаж       30      1350
Кабельканал шириной до 25 мм (гипсокартон)      90      4050
Демонтаж кабелей                                30      9000
-------------------------------------------------------------------------------------------------
Итого                                                   16200
-------------------------------------------------------------------------------------------------
Исполнитель - Васечкин Петр Иванович
-------------------------------------------------------------------------------------------------
Начало работ -  23.05.2016
Конец работ -   29.05.2016
-------------------------------------------------------------------------------------------------
Статус - Не выполнен
Статус оплаты - Не оплачен
Тип оплаты - Наличный расчет
```

Рисунок 1 - Бланк отчета о заказе

Необходимо реализовать автоматическое формирование этих документов для каждого выбранного или вновь сформированного заказа.

Для того чтобы вести правильный учет по всем проведенным сделкам, административный персонал должен получать все необходимые сведения от информационной системы ElectroPlus, в том числе отчеты по любой интересующей информации. На рис. 2 предоставлен список сотрудников фирмы ООО «ЭлектроПлюс».

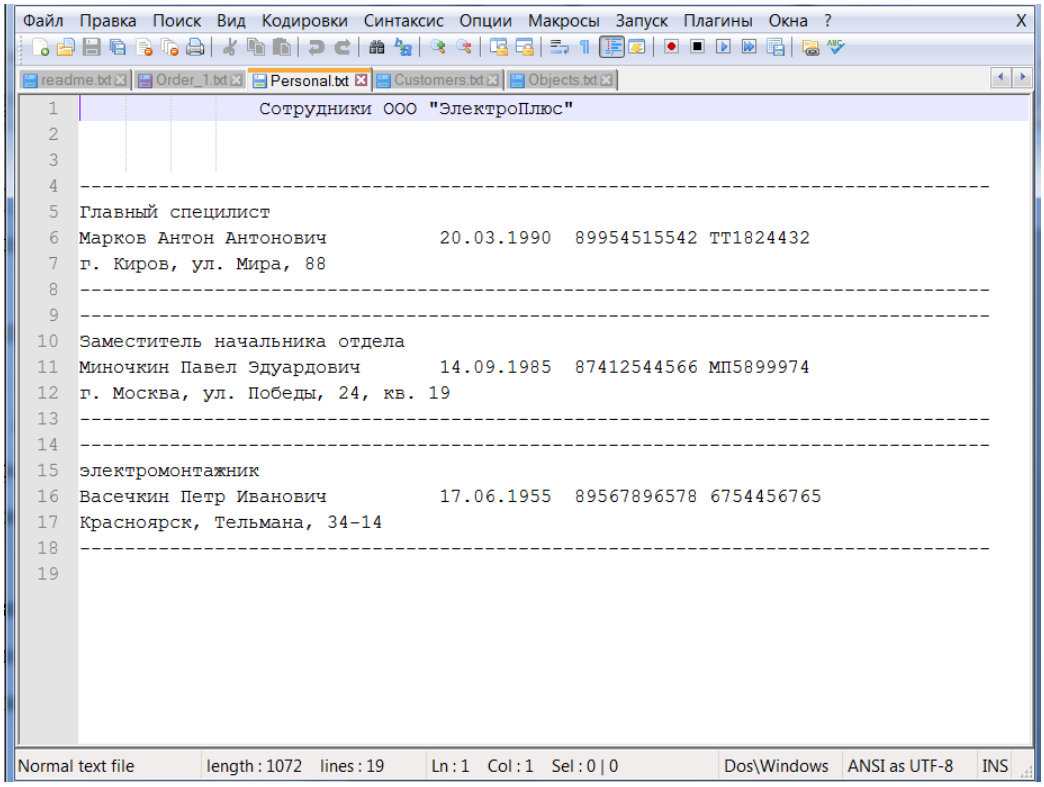


Рисунок 2 - Бланк отчета о сотрудниках фирмы

Система должна иметь защиту от несанкционированного доступа и должна обеспечивать:

- идентификацию пользователей;
- проверку полномочий пользователей при работе с системой;
- разграничение доступа пользователей на уровне задач и доступа к базе данных;

Защищённая часть системы должна использовать "слепые" пароли, при котором символы не показываются на экране, а заменяются точками.

Защищённая часть системы должна автоматически блокировать сессии пользователей и приложений по заранее заданным временам отсутствия активности со стороны пользователей и приложений.

1.3 Проектирование информационной системы

Процесс создания информационной модели начинается с определения концептуальных требований ряда пользователей. Концептуальные требования могут определяться и для некоторых задач (приложений), которые в ближайшее время реализовывать не планируется. Это может несколько повысить трудоемкость работы, однако поможет наиболее полно учесть все нюансы функциональности, требуемой для разрабатываемой системы, и снизит вероятность переделки в дальнейшем. Требования отдельных пользователей должны быть представлены в едином «обобщенном представлении». Последнее называют концептуальной моделью.

Таким образом, концептуальная модель является, по существу, моделью предметной области. При проектировании концептуальной модели все усилия разработчика должны быть направлены в основном на структуризацию данных и выявление взаимосвязей между ними без рассмотрения особенностей реализации и вопросов эффективности обработки. Проектирование концептуальной модели основано на анализе решаемых на этом предприятии задач по обработке данных. Концептуальная модель включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области и выявляемых в результате анализа данных. Имеются в виду данные, используемые как в уже разработанных прикладных программах, так и в тех, которые только будут реализованы.

1.3.1 Описание бизнес-процессов в диаграмме IDEF0

Создание современных информационных систем представляет собой сложнейшую задачу, решение которой требует применения специальных методик и инструментов. Неудивительно, что в последнее время среди системных аналитиков и разработчиков значительно вырос интерес к CASE - технологиям и

инструментальным CASE - средствам, позволяющим максимально систематизировать и автоматизировать все этапы разработки программного обеспечения.

На современном рынке средств разработки ИС достаточно много систем. CASE -средства ERwin и BPwin, разработанные фирмой Computer Assotiations, входят в число лучших на сегодняшний день. CASE -средство верхнего уровня BPwin поддерживает методологии IDEF0 (функциональная модель), IDEF3 (WorkFlow Diagram) и DFD (DataFlow Diagram). Функциональная модель предназначена для описания существующих бизнес-процессов на предприятии. Методология IDEF0 предписывает построение иерархической системы диаграмм - единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция - система разбивается на подсистемы и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности. После каждого сеанса декомпозиции проводится сеанс экспертизы: каждая диаграмма проверяется экспертами предметной области, представителями заказчика, людьми, непосредственно участвующими в бизнес-процессе. Такая технология создания модели позволяет построить модель, адекватную предметной области на всех уровнях абстрагирования[3].

Смоделируем и опишем те бизнес-процессы, которые существуют в предприятии ООО «Электроплюс». Вначале составим контекстную диаграмму (рис. 3), описывающую общее описание системы и ее взаимодействие с внешней средой.

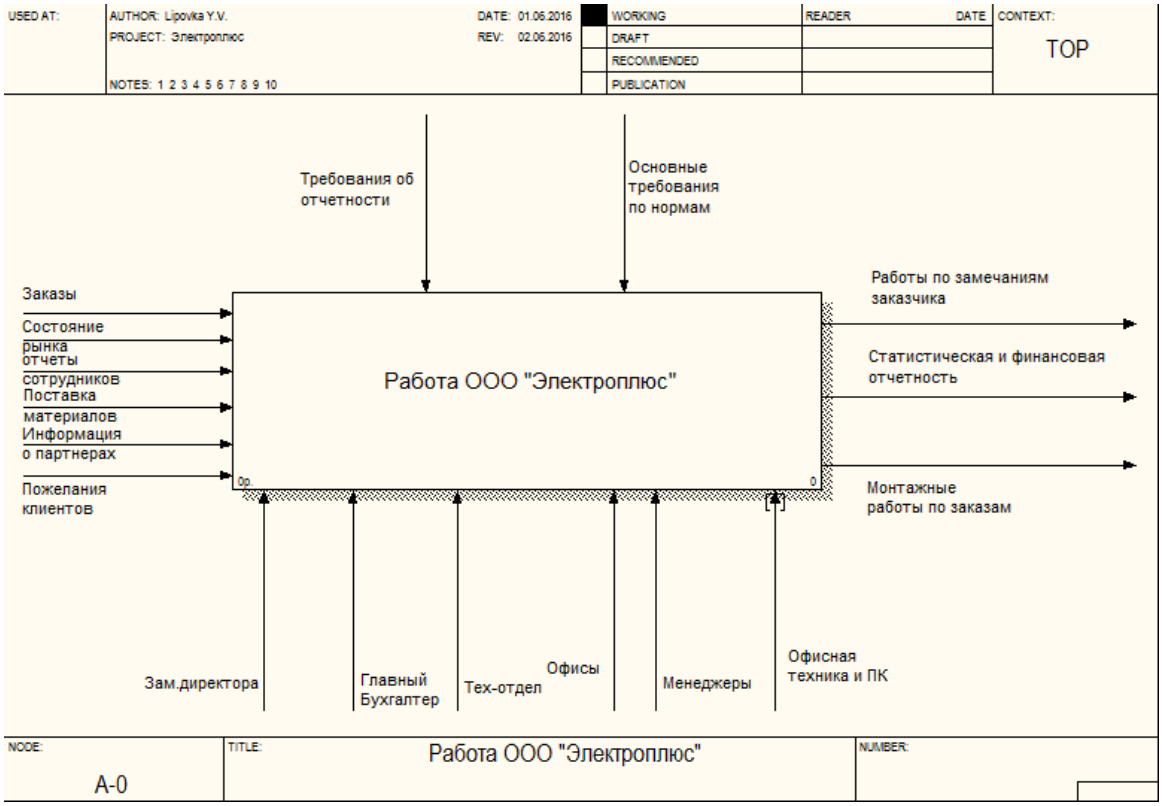


Рисунок 3 - Контекстная диаграмма модели IDEF0

После описания системы проводится функциональная декомпозиция (разбиение системы на крупные фрагменты). Диаграммы, описывающие каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы производится декомпозиция каждого большого фрагмента системы на более мелкие и так далее до достижения нужного уровня подробности описания. Составим декомпозицию контекстной диаграммы (рис. 4).

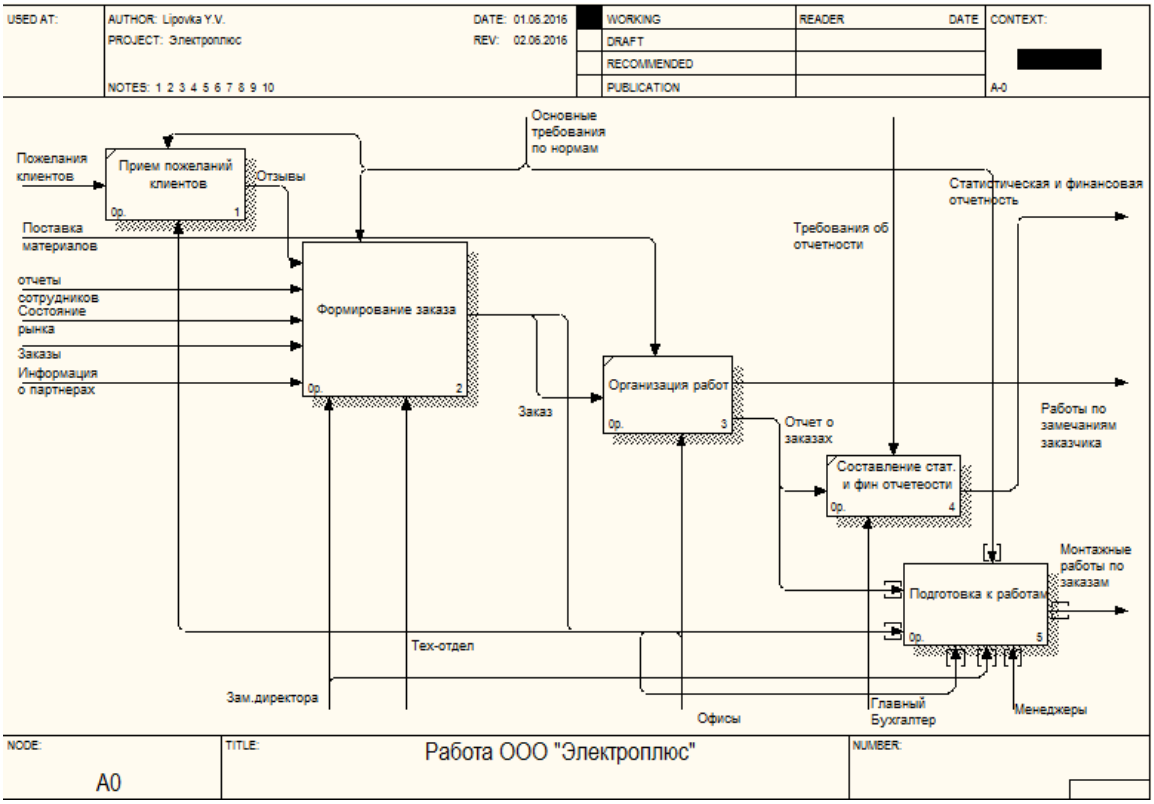


Рисунок 4 - Декомпозиция контекстной диаграммы модели IDEF0

Продолжим и составим декомпозицию узла «Формирование заказа» (рис. 5).

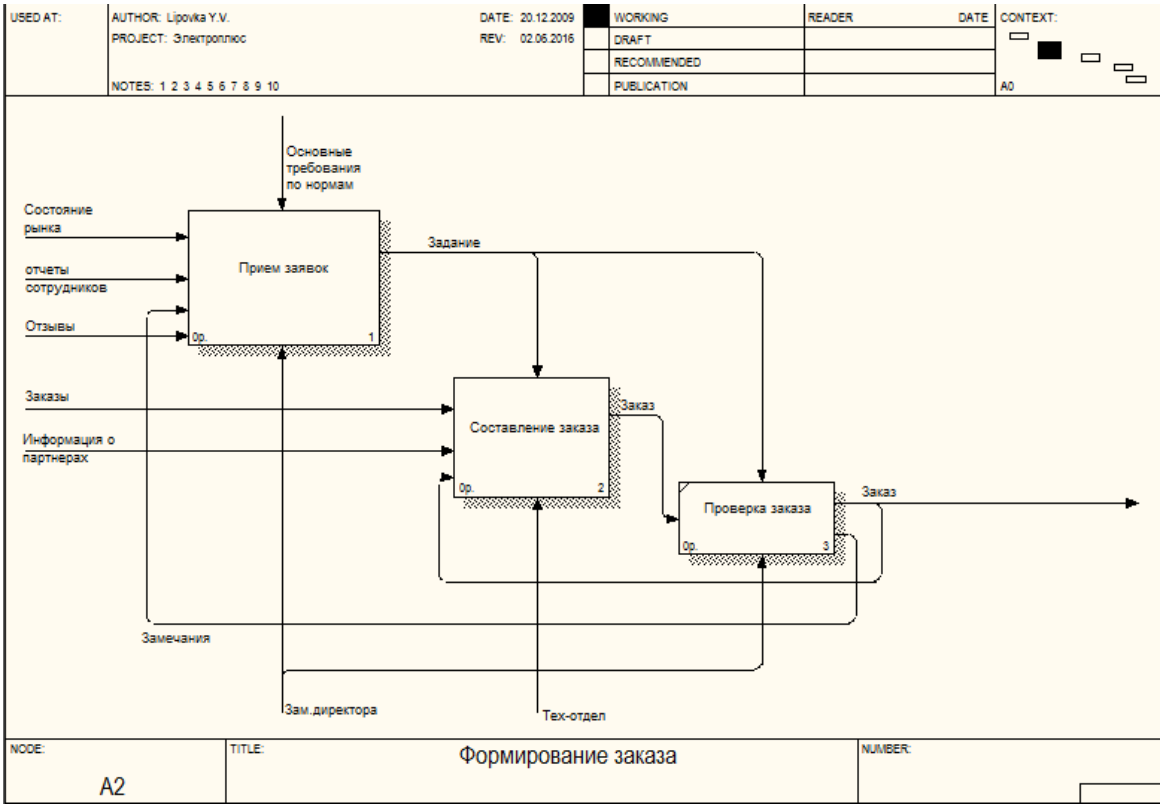


Рисунок 5 - Декомпозиция «Формирование заказа»

Также составим диаграмму дерева узлов, показывающую общую иерархию работ (рис. 6).

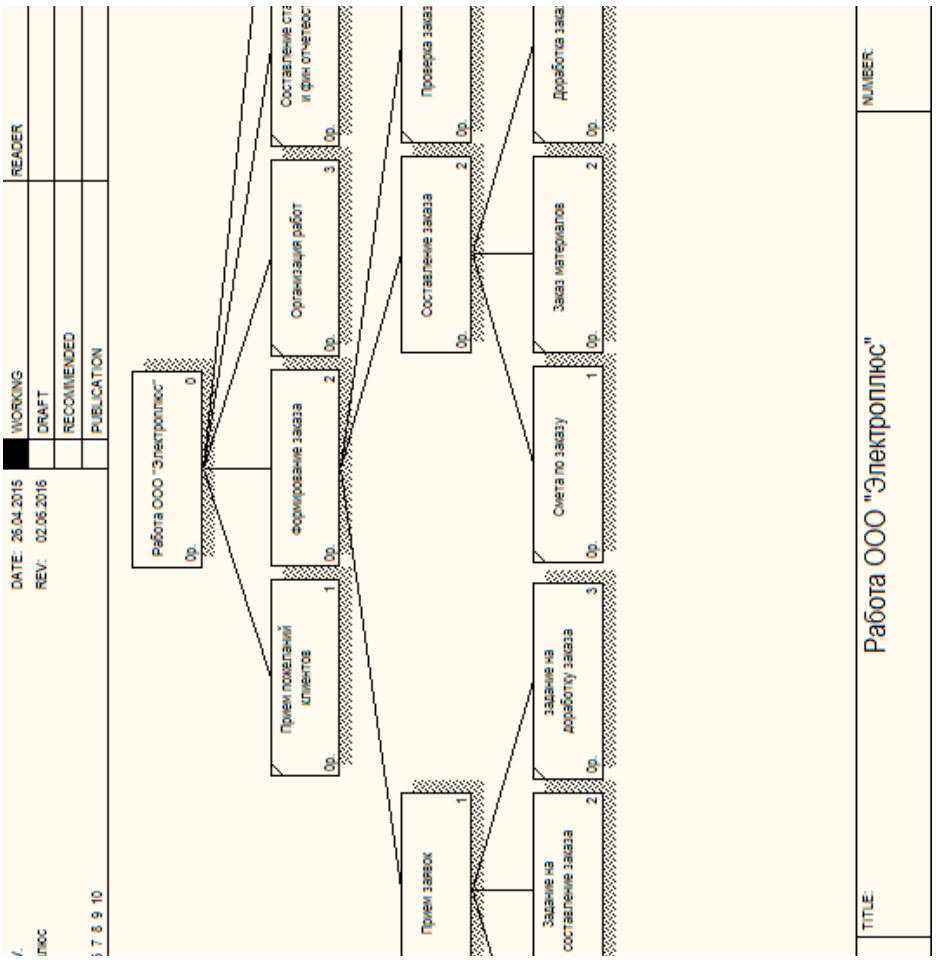


Рисунок 6 - Диаграмма дерева узлов IDEF0 - модели

1.3.2 Инфологическая модель базы данных

Для того чтобы спроектировать полноценную информационную систему, нам не обойтись без создания модели базы данных для нашей системы. Сначала создадим обобщенное описание базы данных с использованием естественного языка, формул, таблиц и графиков. Такая модель, которая полностью независима от физических параметров среды хранения данных, называется инфологической моделью.

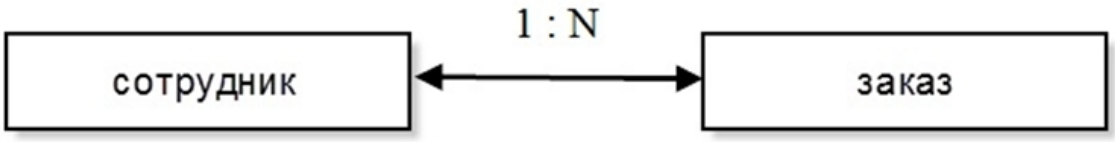
Исходя из анализа предметной области и задач, решаемых системой можно определить следующие типы объектов:

- Сотрудники
- Заказчики
- Объекты
- Услуги
- Заказы

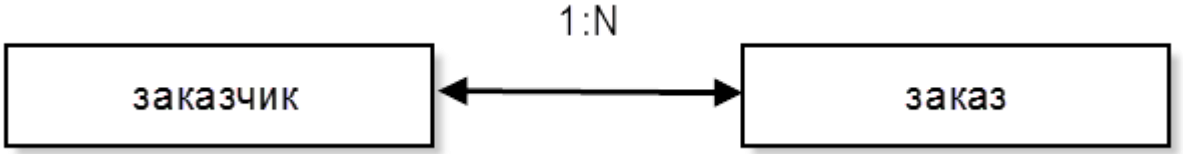
Рассмотрим анализ связей между объектами предметной области. Взаимосвязь выражает отображение или связь между двумя множествами данных. Различают взаимосвязи типа "один к одному", "один ко многим" и "многие ко многим".

Определим для включенных в модель сущностей взаимосвязи. Полученная после этого информационная модель представлена на рисунке 7.

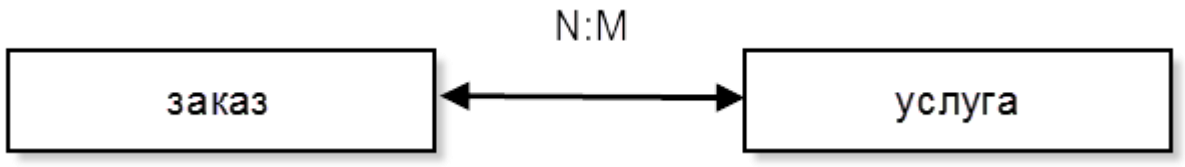
Связь между сущностями «заказ» и «сотрудник» характеризуется сотрудником, который выполняет данный заказ. Хотя один и тот же сотрудник может быть задействован в разных заказах.



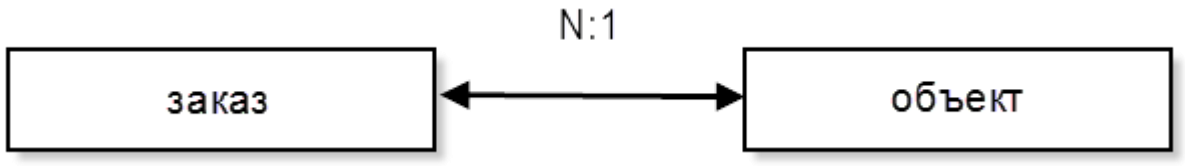
Связь между сущностями «заказ» и «заказчик» характеризуется наличием у заказа заказчика (только одного). Хотя один заказчик может оформить несколько заказов.



Связь между сущностями «заказ» и «услуга» характеризуется одной или несколькими услугами, которые заказывает заказчик. При этом одна и та же услуга может быть в нескольких заказах.



Связь между сущностями «заказ» и «объект» характеризуется объектом (только одним), на который нацелена услуга заказа. Хотя один и тот же объект может быть в нескольких заказах.



При проектировании базы данных следует придерживаться правил нормализации таблиц:

Правило 1: Каждое поле любой таблицы должно быть уникальным.

Правило 2: Каждая таблица должна иметь уникальный идентификатор (первичный ключ), который может состоять из одного или нескольких полей таблицы.

Правило 3: Для каждого значения первичного ключа должно быть одно и только одно значение любого из столбцов данных, и это значение должно относиться к объекту таблицы.

Правило 4: Должна иметься возможность изменять значения любого поля (не входящего в первичный ключ), и это не должно повлечь за собой изменение другого поля [4].

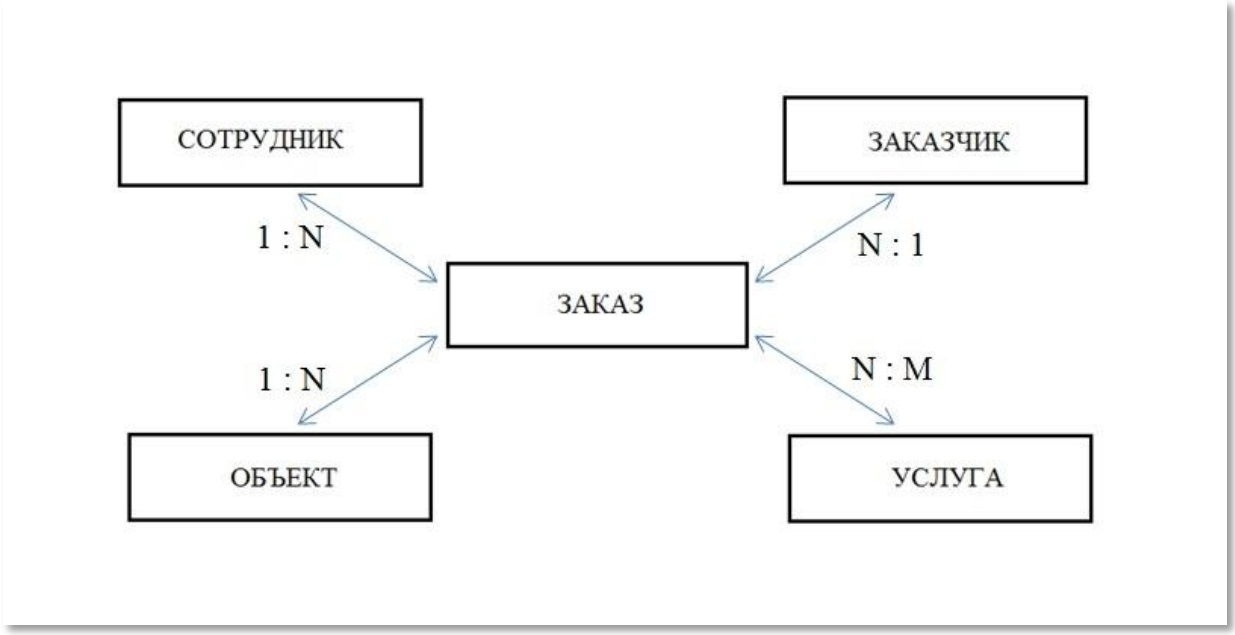


Рисунок 7 - Информационно-логическая модель

Каждый агрегированный объект будет представлен отдельной таблицей базы данных. Элементы данных будут представлены полями таблиц. Имена таблиц и их полей подберем исходя из имен объектов и элементов данных.

В сетевой модели данных понятия главного и подчиненных объектов несколько расширены. Любой объект может быть и главным, и подчиненным (в сетевой модели главный объект обозначается термином «владелец набора», а подчиненный — термином «член набора»). Один и тот же объект может одновременно выступать и в роли владельца, и в роли члена набора. Это означает, что каждый объект может участвовать в любом числе взаимосвязей.

Концептуальная модель транслируется затем в модель данных, совместимую с выбранной СУБД.

1.3.3 Даталогическая модель базы данных, взаимосвязь объектов

Составим схему взаимосвязи объектов на даталогическом уровне (рис. 8).

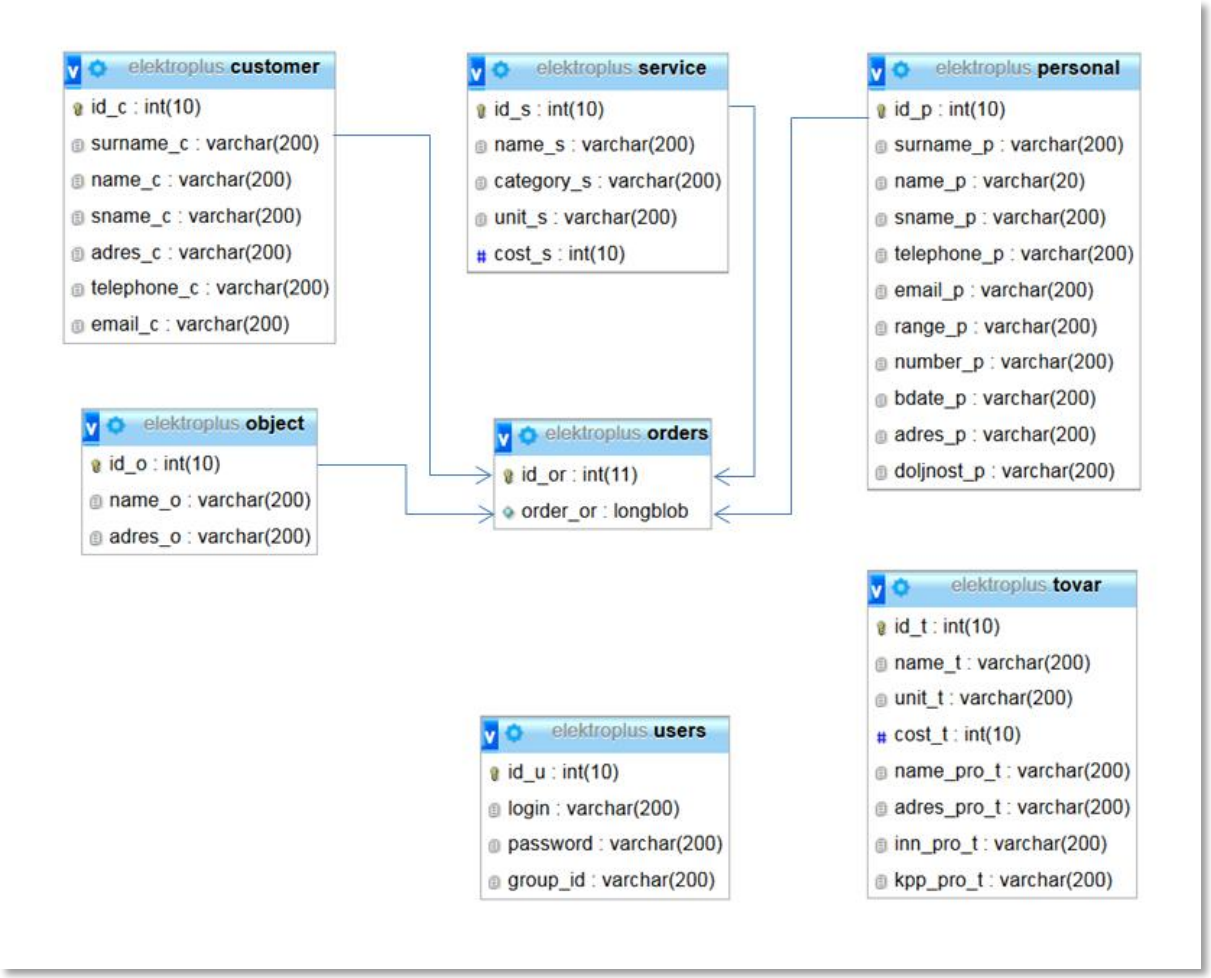


Рисунок 8 – Даталогическая модель БД

Рассмотрим ограничения, которые накладываются на данные. Обеспечением целостности базы данных называется система мер, направленных на поддержание правильности данных в базе в любой момент времени.

В СУБД целостность данных обеспечивается набором специальных предложений, называемых ограничениями целостности. Ограничения целостности - это набор определенных правил, которые устанавливают допустимость данных и связей между ними. Ограничения целостности в большинстве случаев определяются особенностями предметной области.

Ограничения целостности могут относиться к разным объектам БД: атрибутам (полям), записям, отношениям, связям между ними и т. п.

Для полей могут использоваться следующие виды ограничений:

- Тип и формат поля автоматически допускают ввод только данных определенного типа.
- Выбор типа поля Date в формате ДД.ММ.ГГ позволит пользователю ввести только шесть чисел. При этом первая пара цифр не сможет превысить в лучшем случае значения 31, а вторая - 12.
- Задание диапазона значений, как правило, используется для числовых полей. Диапазон допустимых значений может быть ограничен с двух сторон (закрытый диапазон), а может с какой-то одной: верхней или нижней (открытый диапазон).
- Недопустимость пустого поля позволяет избежать появления в БД «ничейных» записей, в которых пропущены какие-либо обязательные атрибуты.
- Задание списка значений позволяет избежать излишнего разнообразия данных, если его можно ограничить.
- Проверка на уникальность значения какого-то поля позволяет избежать записей-дубликатов. Вряд ли будет удобно в справочнике клиентов иметь несколько записей для одного и того же лица.

1.4 Выбор программного обеспечения

1.4.1 Основы построения баз данных

Использование баз данных и информационных систем становится неотъемлемой составляющей деловой деятельности современного человека и функционирования преуспевающих организаций. В связи с этим большую актуальность приобретает освоение принципов построения и эффективного применения соответствующих технологий и программных продуктов: систем управления базами данных, CASE-систем автоматизации проектирования, средств администрирования и защиты баз данных и других.

От правильного выбора инструментальных средств создания информационных систем, определения подходящей модели данных, обоснования рациональной схемы построения базы данных, организации запросов к хранимым данным и ряда других моментов во многом зависит эффективность функционирования разрабатываемой системы. Все это требует осознанного применения теоретических положений и инструментальных средств разработки баз данных и информационных систем.

Базами данных (БД) называют электронные хранилища информации, доступ к которым осуществляется с одного или нескольких компьютеров. Обычно БД создается для хранения и доступа к данным, содержащим сведения о некоторой предметной области, то есть некоторой области человеческой деятельности или области реального мира.

Системы управления базами данных (СУБД) — это программные средства, предназначенные для создания, наполнения, обновления и удаления баз данных. Различают три основных вида СУБД: промышленные универсального назначения, промышленные специального назначения и разрабатываемые для конкретного заказчика. Специализированные СУБД создаются для управления базами данных конкретного назначения — бухгалтерские, складские, банковские и т. д. Универсальные СУБД не имеют четко очерченных рамок применения, они рассчитаны «на все случаи жизни» и, как следствие, достаточно сложны и требуют от пользователя специальных знаний. Как специализированные, так и универсальные промышленные СУБД относительно дешевы, достаточно надежны (отлажены) и готовы к немедленной работе, в то время как заказные СУБД требуют существенных затрат, а их подготовка к работе и отладка занимают значительный период времени (от нескольких месяцев до нескольких лет). Однако в отличие от промышленных заказные СУБД в максимальной степени учитывают специфику работы заказчика (того или иного предприятия), их интерфейс обычно интуитивно понятен пользователям и не требует от них специальных знаний.

Эффективность функционирования информационной системы во многом зависит от ее архитектуры. По своей архитектуре СУБД делятся на одно-, двух- и трехзвенные (рис. 9). В однозвенной архитектуре используется единственное звено (клиент), обеспечивающее необходимую логику управления данными и их визуализацию. В двухзвенной архитектуре значительную часть логики управления данными берет на себя сервер, в то время как клиент в основном занят отображением данных в удобном для пользователя виде. В трехзвенных СУБД используется промежуточное звено — сервер приложений, являющееся посредником между клиентом и сервером БД. Сервер приложений призван полностью избавить клиента от каких бы то ни было забот по управлению данными и обеспечению связи с сервером БД.

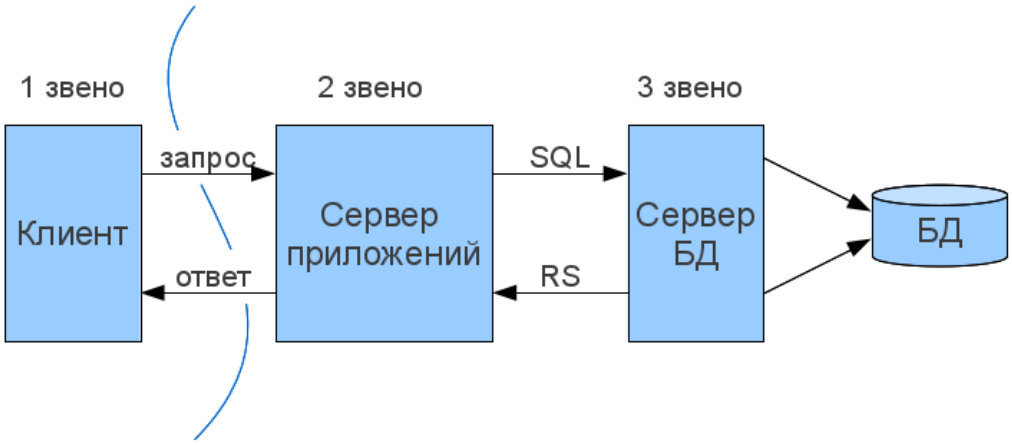


Рисунок 9 - Архитектура СУБД

В зависимости от местоположения отдельных частей СУБД различают локальные и сетевые СУБД.

Все части локальной СУБД размещаются на компьютере пользователя базы данных. Чтобы с одной и той же БД одновременно могло работать несколько пользователей, каждый пользовательский компьютер должен иметь свою копию локальной БД. Существенной проблемой СУБД такого типа является синхронизация копий данных, именно поэтому для решения задач, требующих совместной работы нескольких пользователей, локальные СУБД фактически не применяются. К сетевым относятся файл-серверные, клиент-серверные и

распределенные СУБД. Непременным атрибутом этих систем является сеть, обеспечивающая аппаратную связь компьютеров и делающая возможной корпоративную работу множеств пользователей с одними и теми же данными.

В файл-серверных СУБД все данные обычно размещаются в одном или нескольких каталогах достаточно мощной машины, специально выделенной для этих целей и постоянно подключенной к сети. Такой компьютер называется файл-сервером — отсюда название СУБД. Безусловным достоинством СУБД этого типа является относительная простота ее создания и обслуживания — фактически все сводится лишь к развертыванию локальной сети и установке на подключенных к ней компьютерах сетевых операционных систем. Используемая в дипломном проекте язык программирования C# «умеет» использовать сетевые средства самой популярной в мире ОС — Windows для создания соответствующих клиентских мест, то есть специального программного обеспечения компьютером пользователей. Нетрудно заметить, что между локальными и файл-серверными вариантами СУБД нет особых различий, так как в них все части собственно СУБД (кроме данных) находятся на компьютере клиента. По архитектуре они обычно являются однозвенными, но в некоторых случаях могут использовать сервер приложений. Недостатком файл-серверных систем является значительная нагрузка на сеть[5].

Если, например, клиенту нужно отыскать сведения об одном из заказов, то сети вначале передастся весь файл, содержащий сведения о многих сотнях заказов, и лишь затем в созданной таким образом локальной копии данных отыскивается нужная запись. Ясно, что при интенсивной работе с данными уже нескольких десятков клиентов пропускная способность сети может оказаться недостаточной, и пользователя будут раздражать значительные задержки в реакции СУБД на его требования. Файл-серверные СУБД могут успешно использоваться в относительно небольших фирмах с количеством клиентских мест до нескольких десятков.

Клиент-серверные (двухзвенные) системы значительно снижают нагрузку на сеть, так как клиент общается с данными через специализированного посредника — сервер базы данных, который размещается на машине с данными. Сервер БД принимает запрос от клиента, отыскивает в данных нужную запись и передает ее клиенту. Таким образом, по сети передаются относительно короткий запрос и единственная нужная запись, даже если соответствующий файл с данными содержит сотни тысяч записей. Запрос к серверу формируется на специальном языке структурированных запросов (Structured Query Language, SQL), поэтому часто серверы БД называются SQL-серверами. Серверы БД представляют собой относительно сложные программы, разрабатываемые различными фирмами. К ним относятся, например, Microsoft SQL Server производства корпорации Microsoft, Sybase SQL Server корпорации Sybase, Oracle производства одноименной корпорации, DB2 корпорации IBM и т. д. SQL-сервером является также и сервер InterBase корпорации Borland. Клиент-серверные СУБД масштабируются до сотен и тысяч клиентских мест. Распределенные СУБД могут содержать несколько десятков и сотен серверов БД. Количество клиентских мест в них может достигать десятков и сотен тысяч. Обычно такие СУБД работают на предприятиях государственного масштаба, отдельные подразделения которых разнесены на значительной территории. К таковым, например, относятся подразделения Министерства обороны и Министерства внутренних дел. В распределенных СУБД некоторые серверы могут дублировать друг друга с целью достижения предельно малой вероятности отказов и сбоев, которые могут исказить жизненно важную информацию. Они используют собственные региональные средства связи. Интерес к распределенным СУБД возрос в связи со стремительным развитием Интернета. Опираясь на возможности Интернета, распределенные системы строят не только предприятия государственного масштаба, но и относительно небольшие коммерческие предприятия, обеспечивая своим сотрудникам работу с корпоративными данными на дому и в командировках.

Рассмотрев различные архитектуры СУБД и области их применения, для разработки информационной системы выберем наиболее перспективную и популярную двухзвенную архитектуру клиент-сервер. В достаточно распространенном варианте архитектура клиент-сервер предполагает наличие компьютерной сети и распределенной базы данных, включающей корпоративную базу данных и персональные базы данных. Корпоративная база данных размещается на компьютере-сервере, персональные базы данных размещаются на компьютерах сотрудников подразделений, являющихся клиентами корпоративной БД. В нашем случае при разработке информационной системы будем использовать локальный вариант режима клиент-сервер, при помощи набора программ Denwer, включающую локальный сервер Apache и СУБД MySQL 5.

При использовании технологии клиент-сервер приложение разделяется на две части. Клиентская часть обеспечивает удобный графический интерфейс и размещается на компьютере пользователя. Серверная часть является локальной, находясь на том же самом компьютере пользователя, осуществляет управление данными, разделение информации, администрирование. Клиентское приложение формирует запросы к серверу базы данных, на котором выполняются соответствующие команды. Результат выполнения запроса пересылается клиенту.

1.4.2 SQL - язык структурированных запросов к базам данных

Язык SQL (Structured Query Language - структурированный язык запросов) представляет собой стандартный высокоуровневый язык описания данных и манипулирования ими в системах управления базами данных (СУБД), построенных на основе реляционной модели данных [6].

Язык SQL был разработан фирмой IBM в конце 70-х годов. Первый международный стандарт языка был принят международной стандартизирующей организацией ISO в 1989 г., а новый (более полный) - в 1992 г. В настоящее время

все производители реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL92.

Единственной структурой представления данных (как прикладных, так и системных) в реляционной базе данных (БД) является двумерная таблица. Любая таблица может рассматриваться как одна из форм представления теоретико-множественного понятия отношение (relation), отсюда название модели данных – реляционная.

Стандарт SQL-92 обширен и всеобъемлющ. Ни одна из распространенные коммерческих СУБД, таких как DB2, Oracle или MS SQL Server, не реализует его в полном объеме. Язык SQL ориентирован на текст. Он был разработан задолго до появления графических интерфейсов пользователя, так что для работы с ним требуется лишь текстовый редактор. Разумеется, сегодня в SQL Server, Oracle, DB2 и других СУБД имеются графические средства для выполнения многих из тех задач которые ранее могли быть выполнены только с помощью SQL. Не все из того, что позволяет делать SQL, можно осуществить с помощью графических средств; более того, в ряде случаен, па пример, для динамической генерации операторов SQL в программном коде, SQL использовать необходимо.

С помощью SQL можно определять структуры базы данных, а также запрашивать и обновлять информацию в базе данных.

1.4.3 Выбор Системы Управления Базами Данных (СУБД)

Выбор системы управления баз данных (СУБД) представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранный программный продукт должен удовлетворять потребностям предприятия, при этом должны учитываться финансовые затраты на приобретение оборудования, системы управлениями базами данных, разработку программного обеспечения на ее основе и обучение

персонала. Перечень требований к СУБД может меняться в зависимости от поставленных целей. Выделим общие критерии выбора системы:

- Моделирование данных
- Особенности архитектуры и функциональные возможности
- Контроль работы системы
- Особенности разработки приложений
- Производительность
- Надежность
- Требования к рабочей среде
- Смешанные критерии

Рассмотрим каждую из этих групп в отдельности.

Моделирование данных. Имеется в виду, поддерживает ли СУБД необходимые типы данных, используемые языки запросов.

Особенности архитектуры и функциональные возможности. Является ли СУБД мобильной, т.е. независимой от среды, в которой она работает, распределенной, т.е. поддерживает ли сетевой обмен данными.

Контроль работы системы. Возможность СУБД управлять использованием оперативной и физической памяти.

Особенности разработки приложений. Учитывается, для каких целей разрабатывается АИС, для использования одним, несколькими или тысячами пользователей, будет ли АИС локализовываться в других странах, с другой языковой поддержкой, будет ли это Web-проект или обычная АИС и т.д.

Производительность. Способна ли СУБД распараллеливать процессы обработки запросов, тем самым, понижая время ответа системы на запросы пользователя, предусмотрена ли возможность оптимизации запросов.

Надежность. Понятие надежности системы имеет много смыслов - это и сохранность информации независящая от любых сбоев, и безотказность работы

системы в любых условиях, и обеспечение защиты данных от несанкционированного доступа.

Требования к рабочей среде:

- поддерживаемые аппаратные платформы;
- минимальные требования к оборудованию;
- максимальный размер адресуемой памяти;
- операционные системы, под управлением которых способна работать СУБД.

Смешанные критерии. Такие как, качество и полнота документации, модель формирования стоимости, стабильность производителя, распространенность СУБД.

Даже если просто отмечать, насколько хороши или плохи выделенные параметры в случае каждой конкретной СУБД, то сравнение уже двух различных систем является трудоемкой задачей. Тем не менее, четкий и глубокий сравнительный анализ на основании вышеперечисленных критериев в любом случае поможет рационально выбрать подходящую систему для конкретного проекта, и затраченные усилия не будут напрасными. Перечень критериев поможет осознать масштабность задачи и выполнить ее адекватную постановку.

В дипломном проекте используется СУБД MySQL. MySQL – это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySQL – свободно распространяемая система, т.е. платить за ее применение не нужно. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД, вполне подходящая для не слишком глобальных проектов[7].

Основные преимущества MySQL:

- Многопоточность. Поддержка нескольких одновременных запросов.
- Оптимизация связей с присоединением многих данных за один проход.

- Записи фиксированной и переменной длины.
- ODBC драйвер в комплекте с исходником
- Гибкая система привилегий и паролей.
- До 16 ключей в таблице. Каждый ключ может иметь до 15 полей.
- Поддержка ключевых полей и специальных полей в операторе .
- Поддержка чисел длинной от 1 до 4 байт (ints, float, double, fixed), строк переменной длины и меток времени.
- Интерфейс с языками C и perl.
- Основанная на потоках, быстрая система памяти.
- Утилита проверки и ремонта таблицы (isamchk).
- Все данные хранятся в формате ISO8859_1.
- Все операции работы со строками не обращают внимания на регистр символов в обрабатываемых строках.
- Псевдонимы применимы как к таблицам, так и к отдельным колонкам в таблице.
- Все поля имеют значение по умолчанию. Можно использовать на любом подмножестве полей.
- Легкость управления таблицей, включая добавление и удаление ключей и полей[8].

1.4.4 Выбор платформы программирования

Средства разработки программ работы с БД могут использоваться для создания разновидностей следующих программ:

• клиентских программ;

• серверов БД и их отдельных компонентов;

• пользовательских приложений.

Программы первого и второго вида довольно малочисленны, так как предназначены, главным образом, для системных программистов. Пакетов третьего вида гораздо больше, но меньше, чем полнофункциональных СУБД.

К средствам разработки пользовательских приложений относятся системы программирования, разнообразные библиотеки программ для различных языков программирования, а также пакеты автоматизации разработок (в том числе систем типа клиент-сервер). В числе наиболее распространенных можно назвать следующие инструментальные системы. Delphi и Power Builder (Borland), Visual Basic (Microsoft), Visual C# (Microsoft), SILVERRUN (Computer Advisers Inc.), S-Designer (SDP и Powersoft) и ERwin (LogicWorks).

Для разработки информационной системы «ЭлектроПлюс» выберем систему программирования Microsoft Visual Studio 2015, язык программирования C# - инструмент быстрой разработки клиент-серверных приложений.

Разработка клиентских приложений в языке программирования Microsoft Visual C# для БД реализована чрезвычайно гибко и грамотно (содержит развитые средства взаимодействия с БД, с помощью которых можно осуществлять доступ к практически любым реляционным базам данных). Любая прикладная задача ложится на него легко. Время показало правильность многих заложенных в инструмент решений.

Высокая производительность и поддержка различных серверов баз данных превращают Microsoft Visual C# в идеальное решение для создания систем, использующих серверы баз данных разных производителей, и разработки надежных приложений, способных работать с разнородными серверами баз данных[9].

Microsoft Visual Studio 2015 - интегрированная среда, упрощающая создание, отладку и развертывание приложений. Система включает мощные редакторы и новейшие методы координирования совместной деятельности разработчиков и дизайнеров. Интегрированная поддержка разработки через тестирование и новые инструменты отладки позволяют быстро и без труда находить и устранять ошибки, обеспечивая высокое качество решений.

– Создание приложений для Windows 7 - Visual Studio 2015 включает встроенные инструменты разработки для Windows 7, в том числе такие

компоненты пользовательского интерфейса, как мультисенсорный ввод и лента, которые составляют основу передовой технологии Windows 7.

- Простое создание приложений на базе RIA и WPF - Новая функция привязки данных перетаскиванием (в Windows Presentation Foundation) и конструкторы Silverlight упрощают и ускоряют построение приложений для специалистов по проектированию и разработке.
- Настройка Visual Studio соответственно собственному стилю Основное улучшение IDE - включение поддержки для множества мониторов и повышение четкости текста - делает привычную среду еще более продуктивной.
- Применение разработки через тестирование Visual Studio формирует весь код заглушек, необходимый для выполнения модульного тестирования, позволяя разработчикам сосредоточиться на логике приложения.
- Меньше времени на отладку Встроенная иерархия вызовов позволяет быстро прослеживать поток выполнения программы без вызова отладчика. Также для упрощения отладки можно использовать метки для точек останова.
- Интегрированная система контроля версий, отслеживание дефектов и автоматизация сборки Visual Studio 2015 с MSDN включает Team Foundation Server 2015, который является идеальной системой контроля версий, отслеживания дефектов и автоматизации сборки для пользователей Visual Studio. Базовая установка Team Foundation Server превосходно подходит для использования на настольных компьютерах и для начинающих пользователей, до этого работавших с Microsoft Visual SourceSafe.

1.5 Обзор аналогов систем автоматизации деятельности предприятий

Существует большое количество систем направленных на автоматизацию той или иной деятельности. Одни из них претендуют, на звание комплексных

систем автоматизации, другие не пытаются объять необъятное и специализируются на автоматизации конкретных бизнес процессов.

На Российском рынке можно выделить ряд зарубежных разработчиков средств автоматизаций - единственным минусом которых является недостаточно развитые сети партнёров, а так же то, что все они созданы без учета специфики традиционного российского делопроизводства.

В настоящее время крупные иностранные компании, работающие в сфере информационных технологий, не рассматривают российский рынок в числе ключевых и поэтому неактивно продвигают свои информационные системы.

Исключение составляют лишь несколько компаний.

Среди них американская компания Documentum (ее официальным дистрибьютором в России является "Документум Сервисиз"). Компания продвигает собственный продукт Documentum 4i. Система включает в себя возможность разработки функциональных приложений с помощью специально разработанного для этого модуля Developer Studio – в котором и происходит конфигурирование прикладного решения.

Другая популярная в России зарубежная система — DOCS Open канадской фирмы Hummingbird (официальный дистрибьютор — фирма HBS). DOCS Open дополняют другие программные разработки канадской фирмы: DOCSFusion, CyberDOCS, PowerDOCS, которые объединяются в решение с новым названием Hummingbird DM.

DOCS Open может эффективно применяться и в крупных организациях с большим числом сотрудников (тысячи человек), и в небольших фирмах, где работает пять-шесть человек. Система в первую очередь предназначена для организаций, которые интенсивно занимаются созданием документов и их редактированием (головные офисы компаний, консалтинговые компании, органы власти и т. д.).

DOCS Open - это открытая платформа, к ней поставляются средства разработки для создания специализированных приложений или интеграции с другими системами.

Продукт не ориентирован на применение в области инженерно-конструкторского документооборота. В разобщенных территориально организациях могут возникнуть проблемы, так как в системе нет механизмов репликации информации (т. е. информация, доступная в центре, может быть недоступной в филиалах организации). В ней имеются средства поддержки совместной работы на уровне рабочей группы.

Доля российских разработчиков систем автоматизация достаточно велика, рассмотрим более подробно наиболее популярные отечественные средства разработки прикладных решений.

«Босс-референт». Разрабатывалась компанией «АйТи». Основное применение – создание корпоративных систем, охватывающая деятельность сотрудников на своих рабочих местах. Поддержка делопроизводства, организационное управление, согласование документов. Система реализована на платформе Lotus Notes. Благодаря этому вдобавок к функциям "БОСС-Референт" пользователи получают в свое распоряжение и функции среды Lotus Notes, включая электронную почту, репликацию данных, возможность удаленной работы и т. д.

«БЭСТ-5». Разрабатывается с 2001 года коллективом специалистов по информационным технологиям, бухгалтерскому и налоговому учету. В программе имеют средства для ведения учета и выработки управленческих решений по всем аспектам деятельности современного торгового, производственного или бюджетного предприятия. Для этих целей в программе имеется ряд базовых функциональных блоков: финансы, торговля, производство, персонал. Кроме того, имеется постоянно расширяемый набор отраслевых и специализированных решений (БЭСТ-МАГАЗИН, БЭСТ-АЛКО, БЭСТ-ПИТАНИЕ и др.), которые развивают и дополняют основную программу.

«ИС Парус». Корпоративная информационная система на архитектуре «клиент-сервер». Поддерживает СУБД FoxPro и Oracle. Характеризуется масштабируемостью и модульностью. Поддержка документооборота, анализа данных и интеграции с WEB.

«ИС Алеф». Система Алеф представляет собой конструктор, предназначенный для построения сложных корпоративных аналитических информационных систем. С помощью Системы Алеф решаются задачи ведения оперативного, бухгалтерского и управленческого учета, оперативного планирования и контроля исполнения бюджетов, а также составление различного рода отчетности для всех групп пользователей: бухгалтеров, управляющих, внешних заинтересованных сторон и др.

«1С:ПРЕДПРИЯТИЕ 8.3». Программный продукт, который поставляется с типовыми конфигурациями, которую можно при желании изменить под специфику конкретного учреждения. Создание оригинальных конфигураций позволяет решать с помощью «1С:Предприятия» самые разнообразные задачи по автоматизации экономической деятельности предприятий.

2 Проектная часть

Основной целью построения автоматизированной информационной системы фирмы «ЭлектроПлюс» является хранение информации о деятельности фирмы (ее заказах), о ее сотрудниках и заказчиках.

В любой момент информация о текущем состоянии дел фирмы может быть просмотрена на формах или распечатана из отчетов БД.

Рассмотрим проектную часть разработки информационной системы. В нее включаются разработка таблиц для базы данных, разработка форм приложения, SQL запросы и тестирование системы.

2.1 Разработка программных модулей

2.1.1 Разработка таблиц базы данных

Существует три разновидности связей между таблицами базы данных:

- «один-ко-многим»,
- «один-к-одному»,
- «многие-ко-многим».

Когда одна запись в таблице А может быть связана с 0, 1 или множеством записей в таблице В, вы имеете дело со связью один-ко-многим. В реляционной модели данных связь один-ко-многим использует две таблицы (Рис. 10).

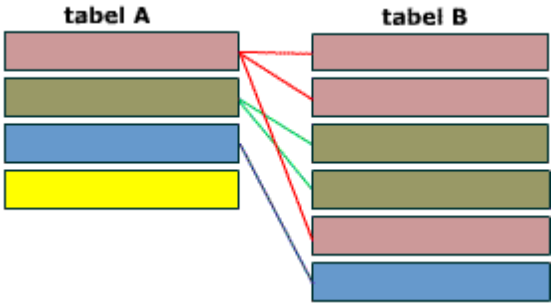


Рисунок 10 - Связь один-ко-многим

Схематическое представление связи один-ко-многим. Запись в таблице имеет 0, 1 или множество ассоциированных ей записей в таблице В.

При связи один-к-одному каждая запись в одной таблице напрямую связана с отдельной записью в другой таблице.

Связь многие-ко-многим – это связь, при которой множественным записям из одной таблицы (А) могут соответствовать множественные записи из другой (В).

Связь многие-ко-многим создается с помощью трех таблиц. Две таблицы – “источника” и одна соединительная таблица. Первичный ключ соединительной таблицы A_B – составной. Она состоит из двух полей, двух внешних ключей, которые ссылаются на первичные ключи таблиц А и В (Рис. 11).

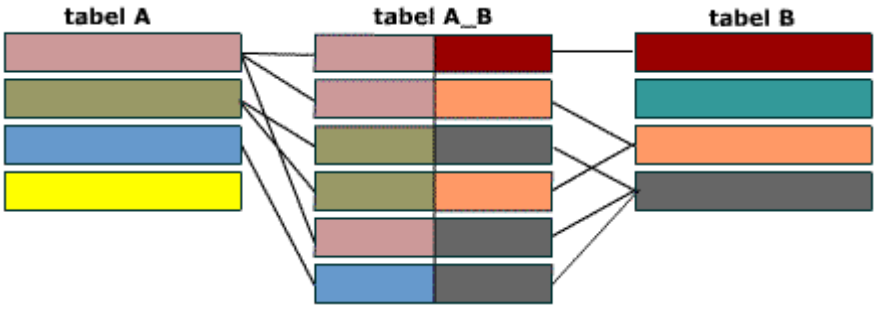


Рисунок 11 - Связь многие ко многим

Все первичные ключи должны быть уникальными. Это подразумевает и то, что комбинация полей A и B должна быть уникальной в таблице A_B.

В нашем случае связь многие-ко-многим это

– Заказ - Услуга (в заказе может быть несколько услуг, так и одна и та же услуга может быть в нескольких заказах)

В данном проекте для упрощения создания базы данных используется бинарная сериализация данных, с помощью класса BinaryFormatter. Вся информация из таблиц, используемых в заказе сериализуется в бинарный поток и сохраняется в строку longblob таблицы Orders. При выводе данных происходит обратный процесс десериализации и данные структурируются обратно из потока данных[10].

Таблицы базы данных:

Таблица Заказы (рис. 12).

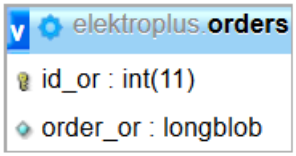


Рисунок 12 - Таблица заказы

Поля таблицы Заказы:

- id_or идентификатор заказа
- order_or бинарный пакет данных заказа

Таблица Сотрудники (рис. 13).

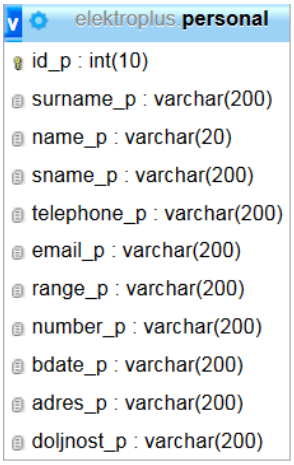


Рисунок 13 - Таблица Сотрудники

Поля таблицы Сотрудники:

- id_p идентификатор сотрудника
- Фамилия
- Имя
- Отчество
- Телефон
- Эл почта
- Серия паспорта
- Номер паспорта
- Дата рождения
- Адрес
- Должность

Таблица Объекты (рис. 14).

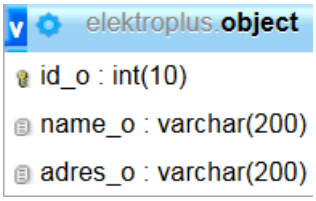


Рисунок 14 - Таблица Объекты

Поля таблицы объекты:

– Id_o –идентификатор объекта

– Название объекта

– Адрес объекта

Таблица Услуги (рис. 15).

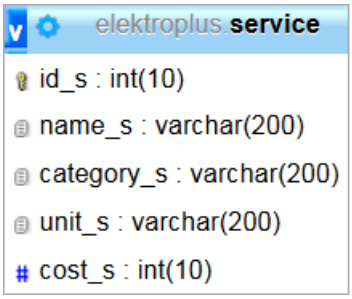


Рисунок 15 - Таблица Услуги

Поля таблицы Услуги:

- Id_s идентификатор услуги
- Название услуги
- Категория услуги
- Единица измерения
- Стоимость в руб за единицу

Таблица Заказчики (рис. 16).

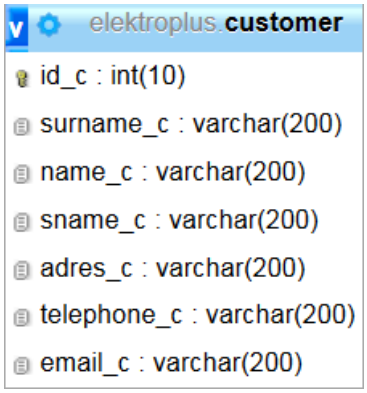


Рисунок 16 - Таблица Заказчики

Поля таблицы Заказчики:

- id_c идентификатор заказчика
- Фамилия
- Имя
- Отчество
- Адрес жительства
- Телефон
- Емайл

Таблица Пользователи (рис. 17).

Рисунок 17 - Таблица Пользователи

Поля таблицы Пользователи:

- id_u идентификатор пользователя
- логин
- пароль
- тип пользователя (сотрудник или клиент)

Таблица Товар (рис. 18).

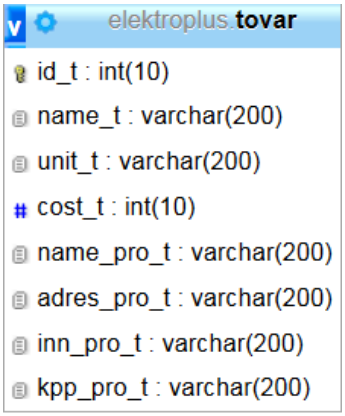


Рисунок 18 - таблица Товар

Поля таблицы Товар:

- id_t идентификатор товара
- название товара
- единица измерения товара
- стоимость за единицу товара (руб)
- название поставщика
- адрес поставщика
- ИНН поставщика
- КПП поставщика

Структура связей в таблицах, разработанная в дизайнере PhpMyAdmin, изображена на рисунке 19.

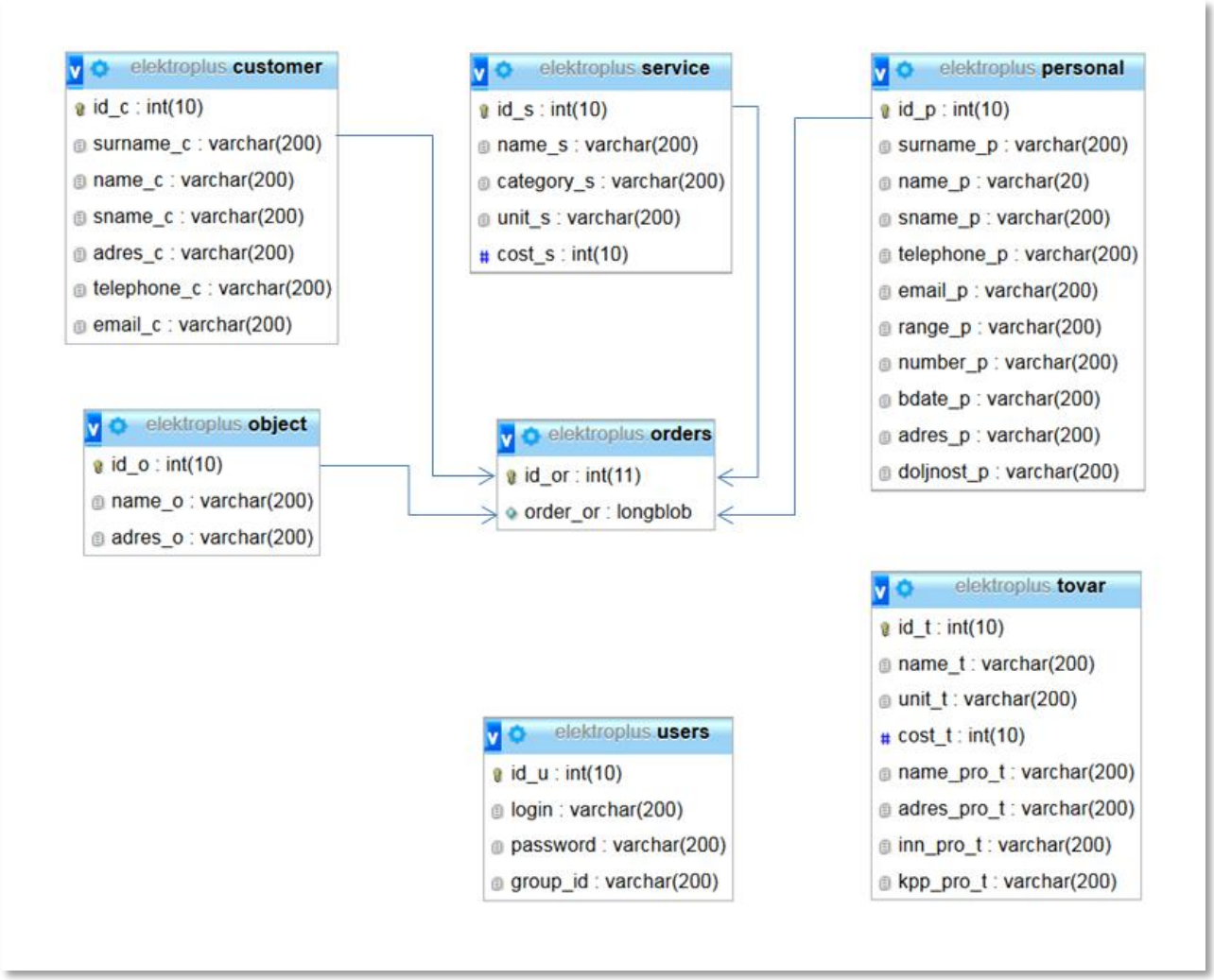


Рисунок 19 - Структура связей в таблицах БД

2.1.2 Разработка форм приложения

ФОРМЫ Windows (также называются WinForms) создают GUI (англ. graphical user interface, графический пользовательский интерфейс) для различных программ. Форма - это графический элемент, появляющийся на рабочем столе. Форма может быть диалоговым окном, просто окном или окном MOI (окно многодокументного интерфейса). Общий процесс проектирования приложений в Windows требует создания формы Windows с указанием ее свойств, с добавлением элементов управления со своими свойствами и реализацией обработчиков событий[11].

Для осуществления загрузки разработанного приложения пользователь запускает программу «ЭлектроПлюс». Для просмотра и редактирования таблиц базы данных созданы формы с одноимёнными названиями.

Каждая форма содержит все необходимые инструменты для работы с таблицей. Реализация функциональности работы с таблицами включает в себя следующее:

– Добавление новых записей в таблицы (под учетной записью администратора);

– Добавление новых записей в таблицу «Orders» (под учетной записью администратора);

– Редактирование записей в таблицах (под учетной записью администратора);

– Удаление записей в таблицах (под учетной записью администратора).

Кроме того, из главной формы возможна загрузка файла справки, предназначенного для объяснения пользователям правил работы с приложением.

В дипломном проекте используются 6 форм:

1. **Autorization** – форма, на которой осуществляется аутентификация пользователя. В случае успешного входа загружается главная форма MainForm;
2. **Error** – форма, выводящая сообщения об ошибках в системе, при вводе данных или некорректных данных при авторизации пользователей.
3. **MainForm** – главная форма, в которой возможен просмотр и изменение всех данных и параметров системы. С помощью данной формы возможен вызов дополнительных форм;
4. **OrderForm** – форма, позволяющая выводить параметры заказа, изменять их (только под учетной записью администратора) и формировать документы на сохранение и печать;

5. **ProductForm** – форма, выводящая перечень товаров и продукции, используемых предприятием.
6. **UserForm** – форма, которая реализует интерфейс для клиентов. В ней отображается прайс-лист на услуги компании.

На рисунке 20 представлена схема взаимодействия форм и передачи данных от форм к базе данных и наоборот.

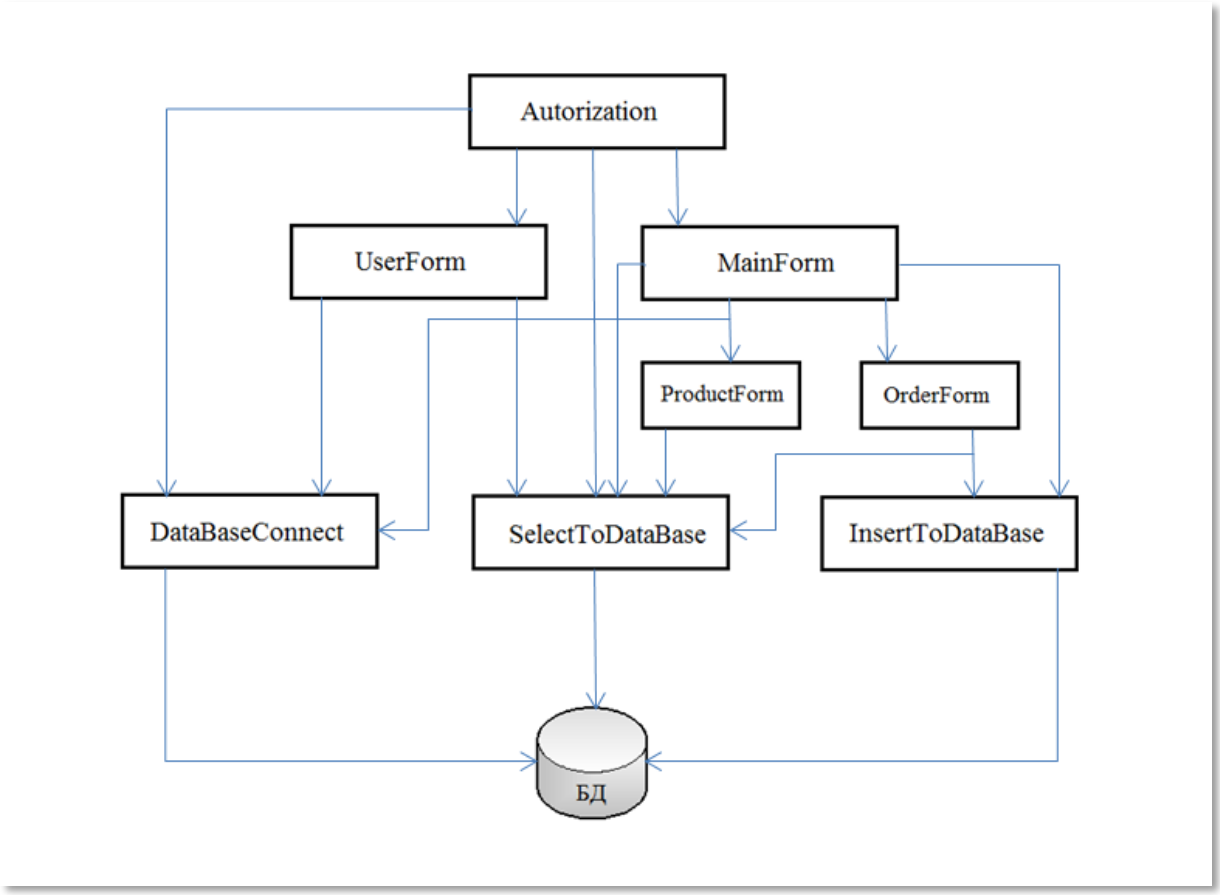


Рисунок 20 - Схема передачи данных от форм к БД

2.1.3 Разработка SQL запросов

Запросам присущ обширный круг функций. С помощью запросов можно просматривать, анализировать и изменять данные из одной и даже нескольких таблиц. Запросы позволяют также обновить или удалить одновременно несколько записей, выполнить встроенные или специальные вычисления. Они также

используются в качестве источника данных для форм и отчетов. Но в первую очередь запросы предназначены для отбора данных на основании критериев.

Оператор SELECT без преувеличений является сердцем SQL. Он позволяет выполнять поиск и предоставлять данные самыми разными способами[12].

Для добавления, изменения, удаления записей в таблицы БД разработаны SQL запросы, которые объединены в следующие классы: InsertToDataBase и SelectToDataBase .

Их можно разбить на группы:

- Запросы для поиска данных, Например запрос поиска заказа по номеру договора:

```
command.CommandText = "SELECT * FROM orders";
 //классу для команд присваиваем переменную с запросом
        command.Connection.Open();
        MySqlDataReader reader;
//объявление переменной класса для считывание информации из БД
        try
        {
           reader = command.ExecuteReader();              //резрешение чтения
           //---- цикл считывания информациии из БД
           while (reader.Read())
           {
              orders.Add((byte[])reader["order_or"]);
           }
           reader.Close();    //прекращение чтения
```

- Запросы для добавления данных, Например запрос добавления заказчика:

```
INSERT INTO customer(surname_c,name_c,sname_c,adres_c,telephone_c,email_c) VALUES('" + surname + "','" + name + "','" + sname + "','" + adres + "','" + telephone + "','" + email + "')"
```

- Запросы для получения данных, Например запрос для получения перечня объектов:

```
command.CommandText = "SELECT * FROM object";
 //классу для команд присваиваем переменную с запросом
          command.Connection.Open();
          MySqlDataReader reader;
//объявление переменной класса для считывание информации из БД
          try
          {
             reader = command.ExecuteReader();              //резрешение чтения
             //---- цикл считывания информациии из БД
             while (reader.Read())
             {
                objects.Add(new                WObject(Convert.ToInt32(reader["id_o"]),
Convert.ToString(reader["name_o"]), Convert.ToString(reader["adres_o"])));
             }
             reader.Close();    //прекращение чтения
```

2.2 Тестирование и отладка информационной системы

После разработки информационной системы необходимо провести тестирование для проверки ее работоспособности и выявления ошибок.

При тестировании системы проверяется работа с формами, редактирование записей в базах данных, корректная работа всех кнопок, выявляются недочеты, которые могут возникнуть в процессе работы с системой. В результате тестирования ошибки не найдены, все мелкие недочеты устранены.

При тестировании проверялись следующие функции:

- ввод новой информации о заказчиках;
- поиск заказа по номеру договора;
- просмотр и изменение данных об услугах;

- получение отчетных документов;

При попытке ошибочного ввода пароля или логина система выдает сообщение, приведенное на рисунке 21.

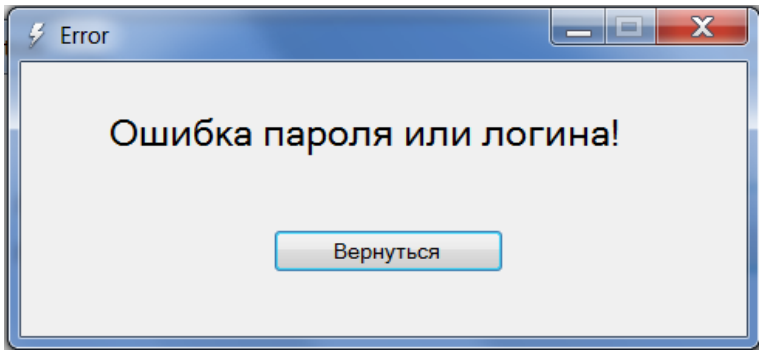


Рисунок 21 - Ошибка ввода пароля или логина

При ошибочном вводе данных услуг в заказе на форме или отсутствии данных в услуге, система выдает ошибку, изображенную на рис. 22.

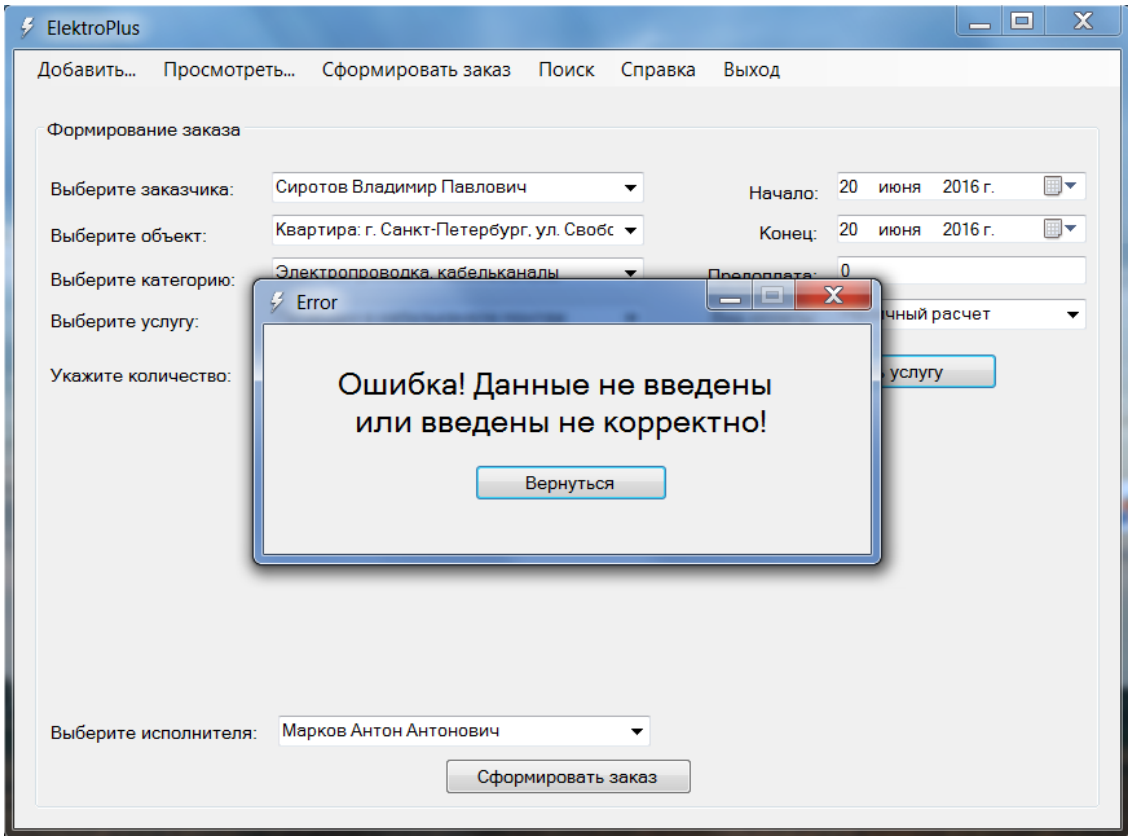


Рисунок 22 - Ошибочный ввод данных

При попытке сформировать пустой заказ при не добавленной услуге, система также выдаст ошибку, что услуги отсутствуют.

Смотрите рисунок 23.

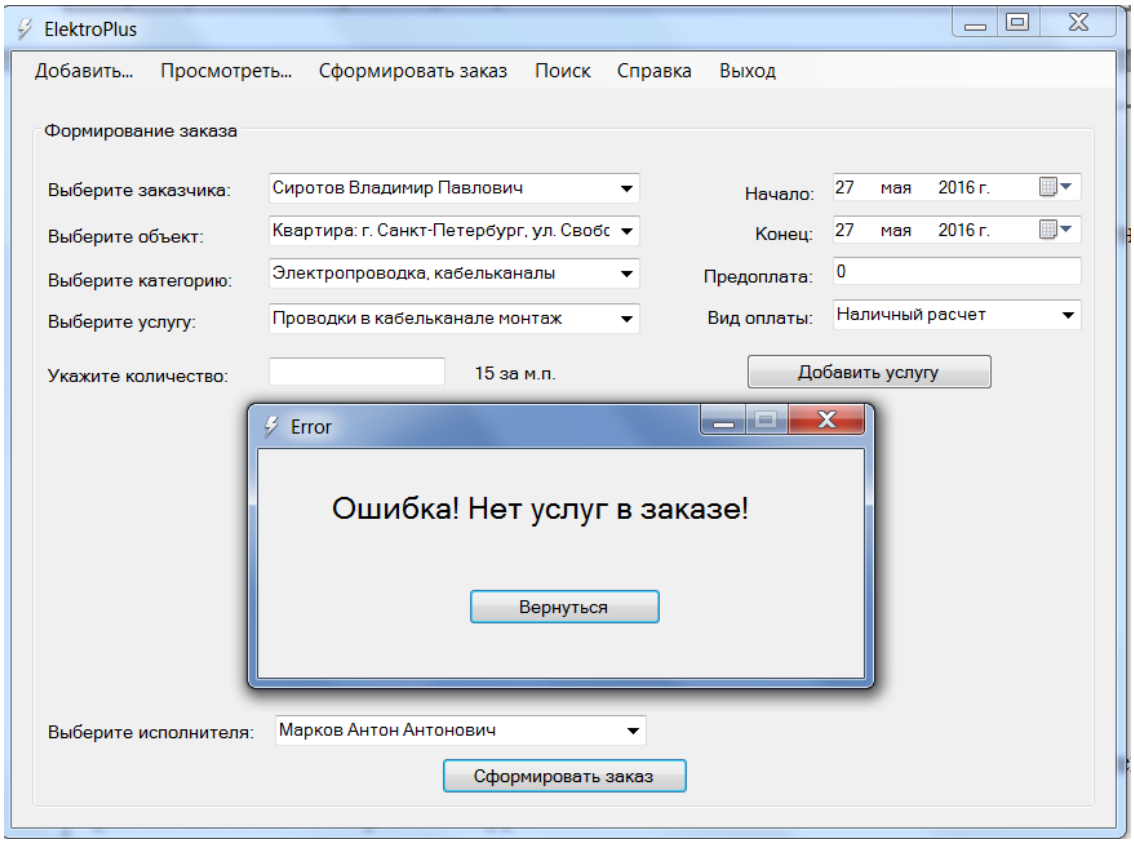


Рисунок 23 - Ошибка. Отсутствие услуг в заказе

Проведенное тестирование показывает работоспособность информационной системы и корректность ее работы. В процессе тестирования были выявлены и устранены мелкие недочеты, которые не нарушают общую работу программы.

3 Описание работы приложения

3.1 Авторизация пользователей

Для запуска программы необходимо запустить приложение «ElektroPlus.exe». После запуска приложение загрузится заставка (рис. 24).

Рисунок 24 - Заставка приложения

Когда заставка исчезнет через 3 секунды, появляется форма авторизации пользователей, в которой нужно ввести соответствующий логин и пароль для соединения с базой данных (сотрудника или клиента) (рис. 25).

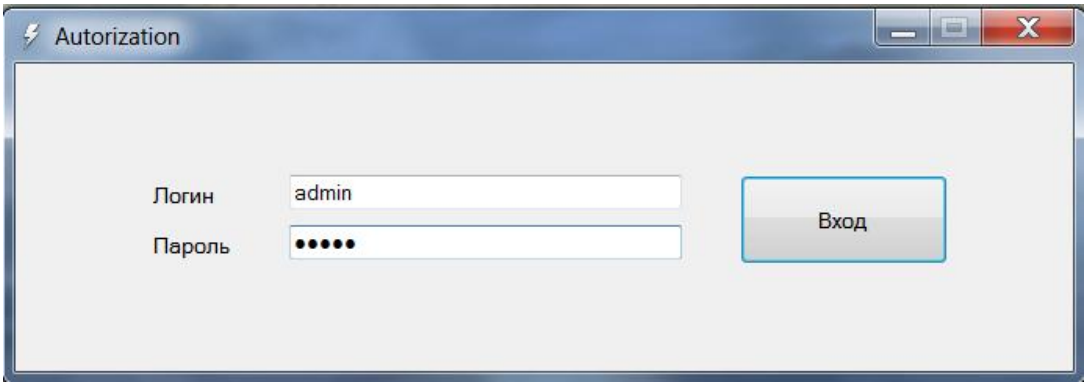


Рисунок 25 - Подключение к базе данных

Затем загружается основная форма сотрудника (рис. 26), если был введен соответствующий логин, откуда доступен просмотр заказчиков, заказов, добавление новых услуг, сотрудников, заказчиков и формирование новых заказов.

Также возможен поиск по заказам и сохранение документов в текстовый файл.

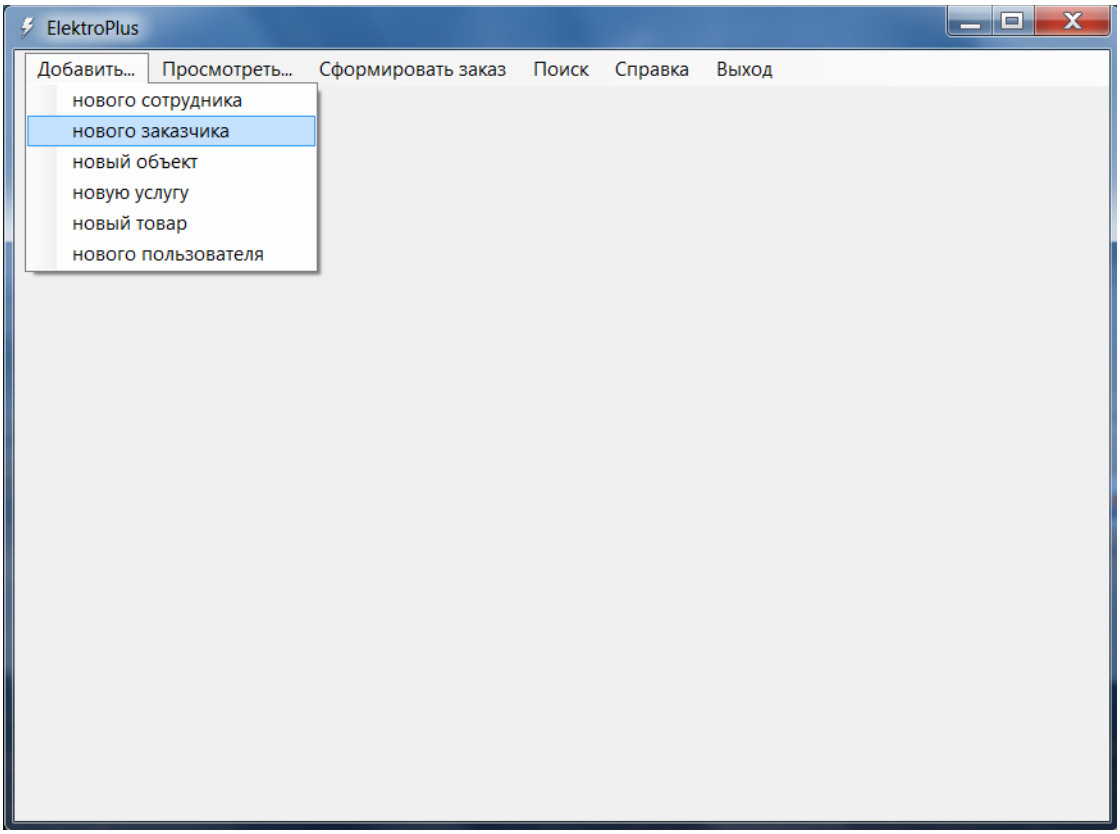


Рисунок 26 - Основная форма приложения для сотрудников

Если в начальной форме авторизуется клиент, то он входит в форму для клиентов, где может просмотреть прайс-лист услуг, разбитых на категории, с возможностью сохранить их в файл и распечатать (рис. 27).

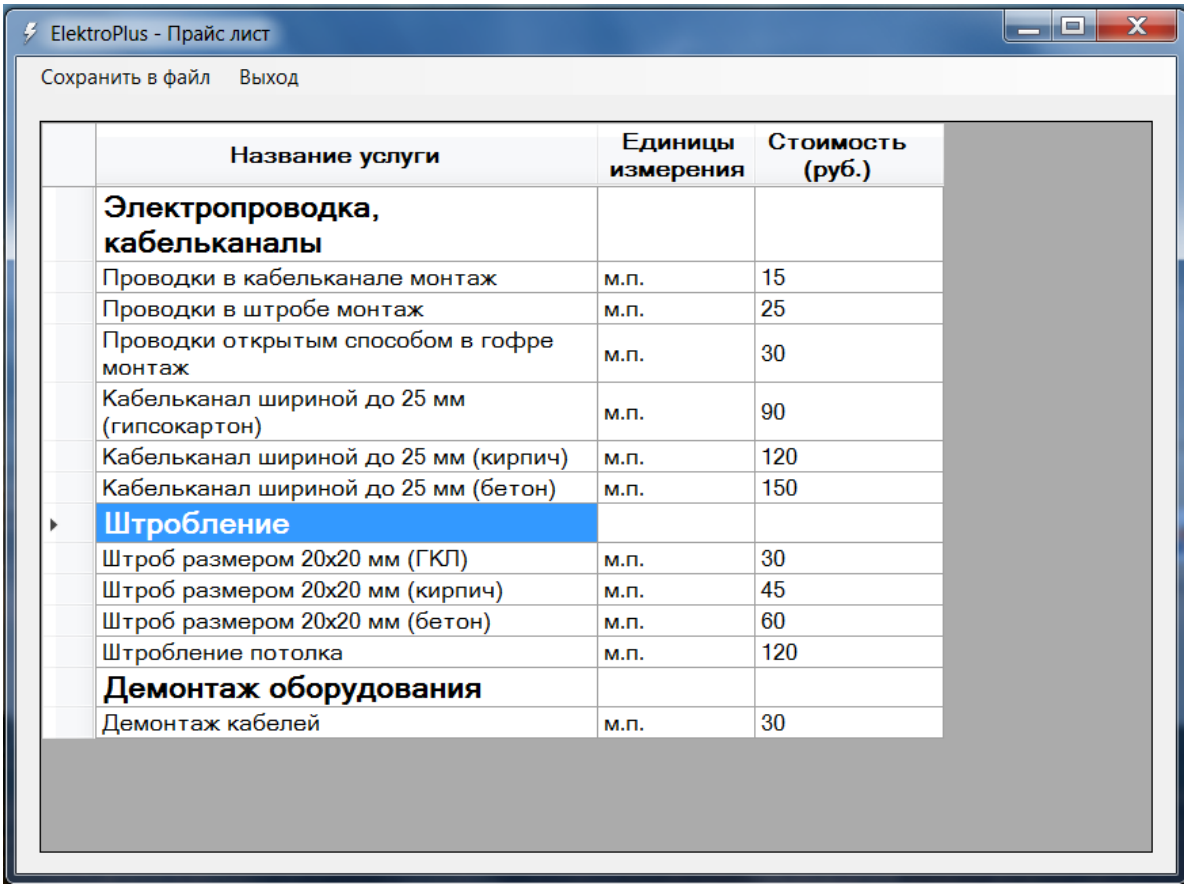


Рисунок 27 - Основная форма для клиентов

3.2 Описание основных программных модулей

Программа предусматривает добавление и редактирование новых заказчиков в базу данных, новых сотрудников, объекты, услуги и товары. К примеру, чтобы добавить в систему нового заказчика, нужно через меню «Добавить…» выбрать соответственно «нового заказчика», тогда появится форма с вводом данных (рис. 28).

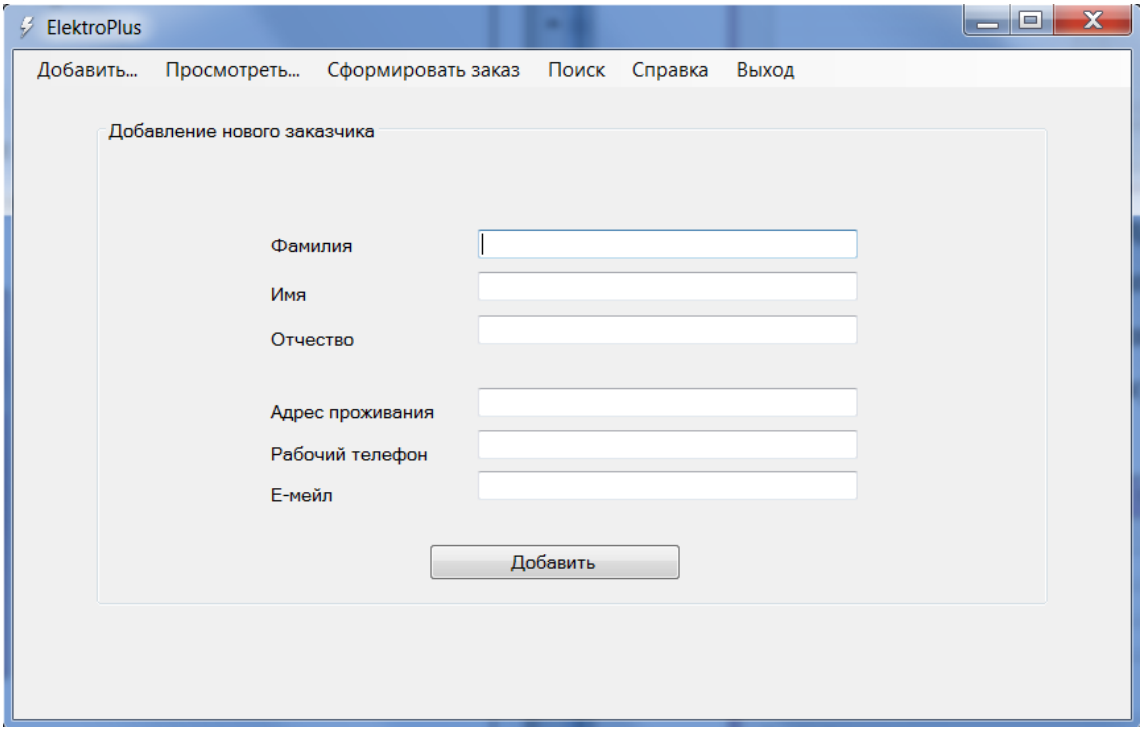


Рисунок 28 - Добавление заказчика

На этой форме вводятся основные данные о заказчиках, когда данные заполнены, необходимо нажать кнопку «Добавить». Если одно или нескольких обязательных полей не введены, то заказчик не добавится, а на экран будет выведена ошибка « Ошибка! Данные не введены или введены некорректно!»

Чтобы добавить новый объект в базу, необходимо через меню «Добавить…» выбрать «новый объект», тогда появляется форма для заполнения новой информации об объекте, на котором будут выполняться заказ. Нужно будет указать название объекта и его адрес (рис.29).

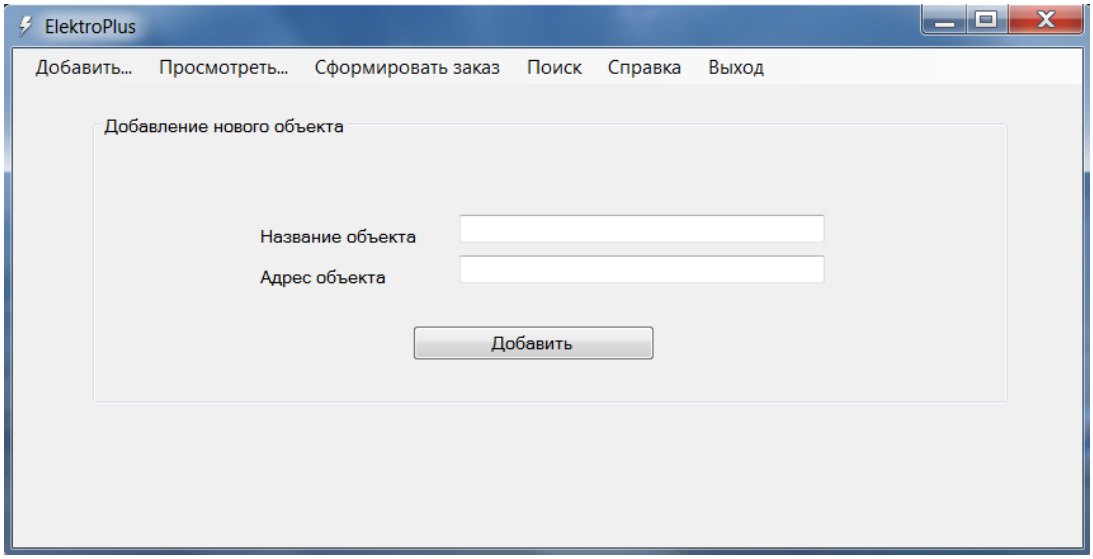


Рисунок 29 - Добавление нового объекта

Если поле "Адрес объекта" или «Название объекта» не введено, то объект не добавится, а на экране появится сообщение об ошибке.

Для того, чтобы сформировать новый заказ, необходимо выбрать верхнее меню «Сформировать заказ», выбрать заказчика и объект из выпадающего списка, затем выбрать категорию услуг, соответствующую услугу и указать количество, которое требуется для заказа, после этого жмем «Добавить услугу» (рис. 30).

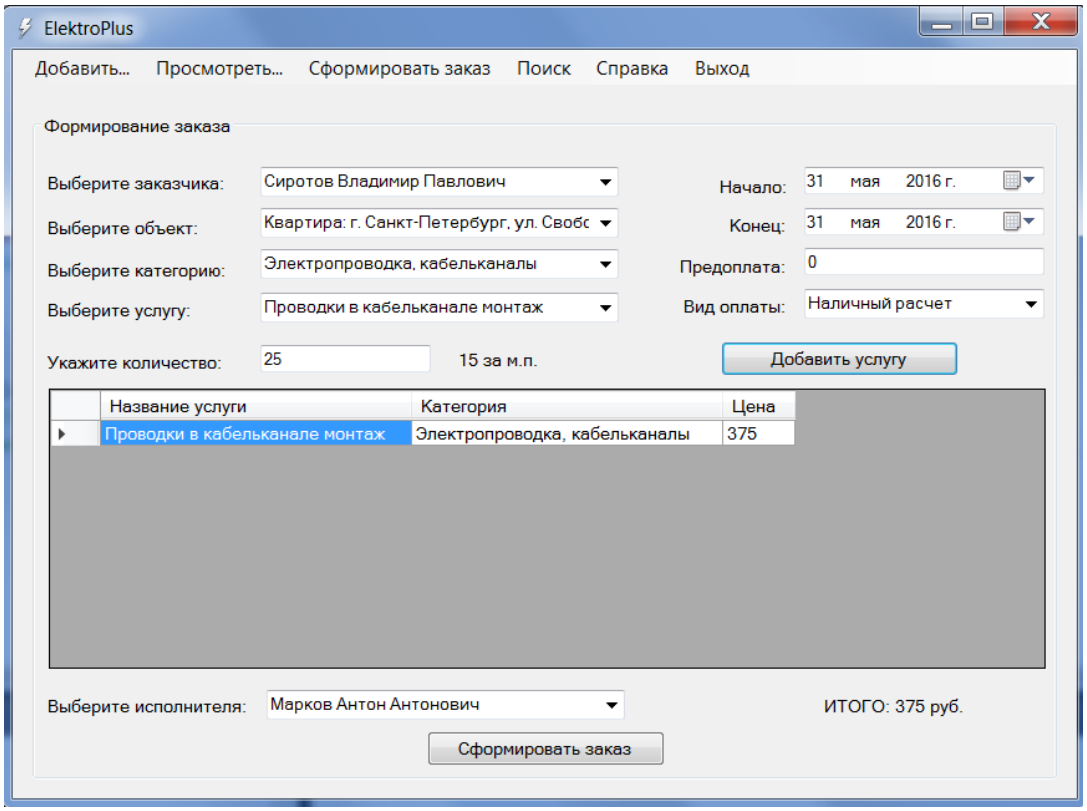


Рисунок 30 - Добавление услуги

Таким же образом добавляем все необходимые услуги, которые нужны для данного заказа. Если каких-то услуг или категорий нет, их необходимо добавить в базу через меню «Добавить…». Отмечаем начало и конец предполагаемых работ, указываем предоплату, если есть и вид оплаты (наличный или безналичный расчет). Итоговая сумма по услугам рассчитывается автоматически, указывается внизу. Выбираем исполнителя работ из списка и жмем кнопку «Сформировать заказ».

Для того чтобы просмотреть сформированный заказ, выбираем в меню «Просмотреть…» подменю «перечень заказов», (рис. 31). В этом подменю находится список всех заказов в базе данных, которые были сформированы. Отсюда можно открыть любой заказ для просмотра, сохранения в файл и печати (рис. 32).

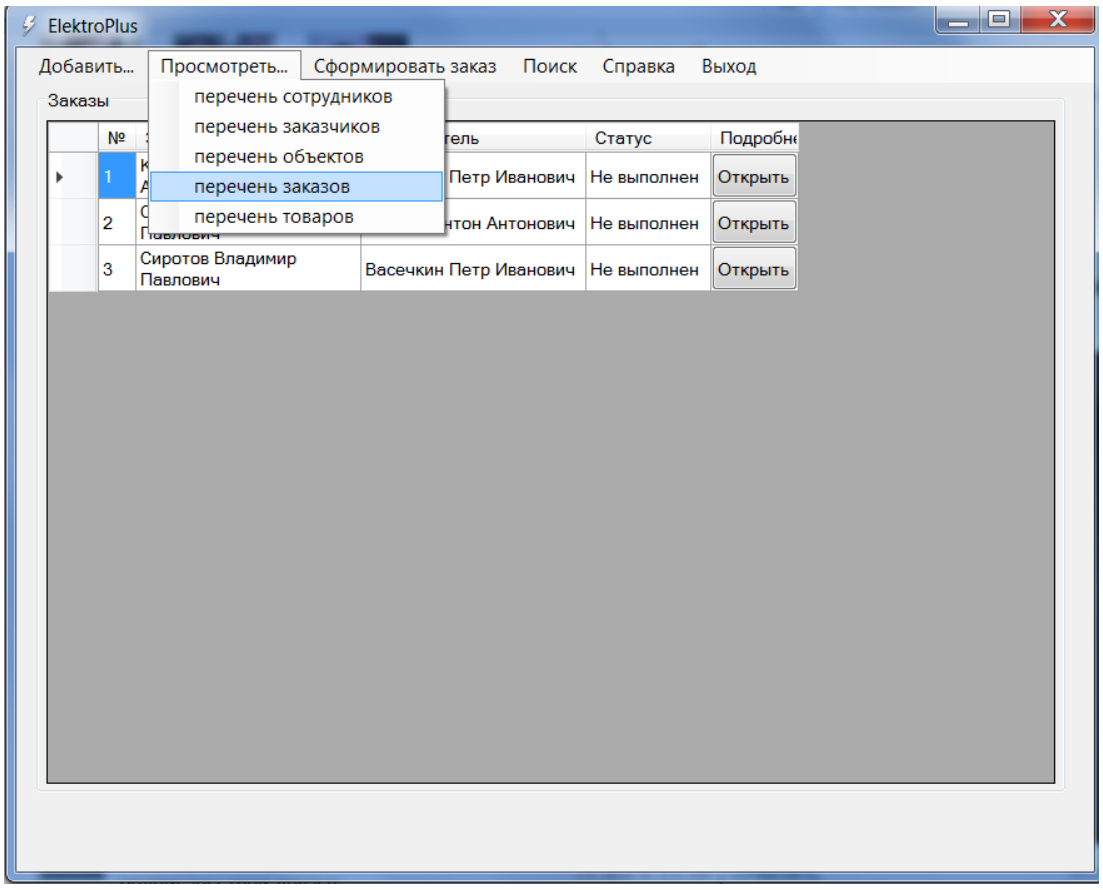


Рисунок 31 - Просмотр заказов в системе

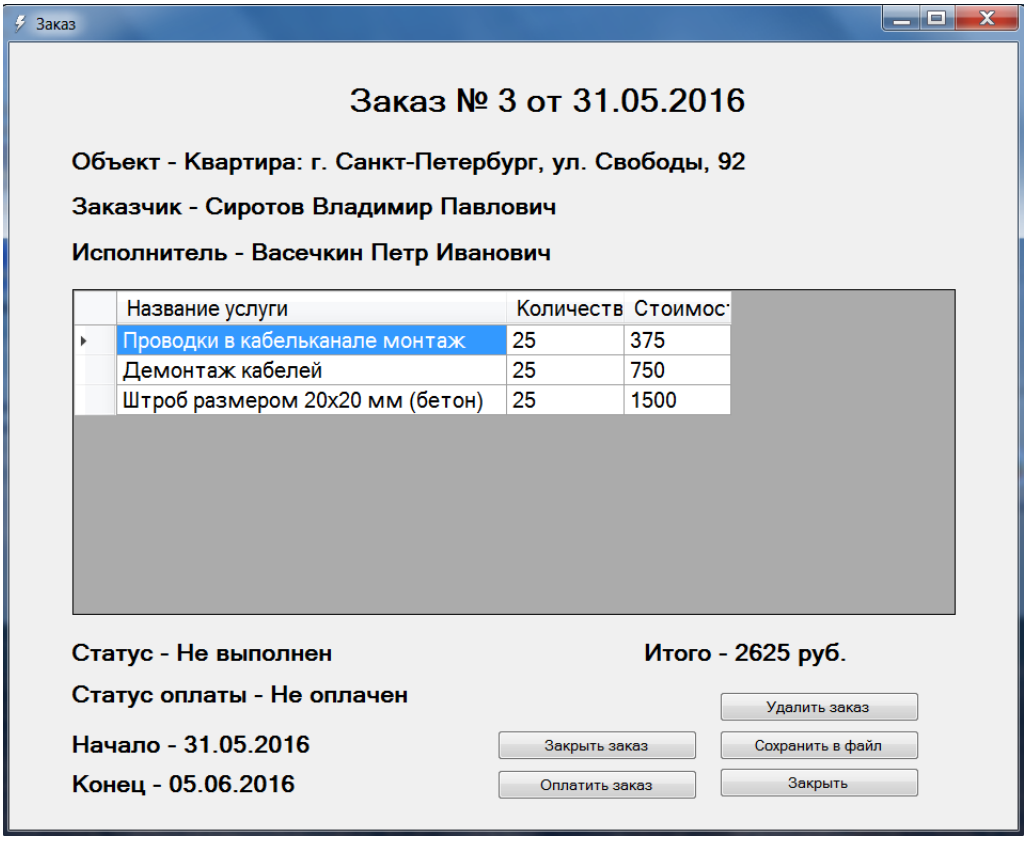


Рисунок 32 - Форма заказа

Если заказ выполнен и оплачен, можно нажать соответствующие кнопки и произойдет изменение статуса заказа на «Выполнен» и «Оплачен». При нажатии кнопки «Сохранить в файл», заказ будет сохранён в txt файл (рис. 33), который можно будет распечатать или сохранить на жесткий диск или флешку.

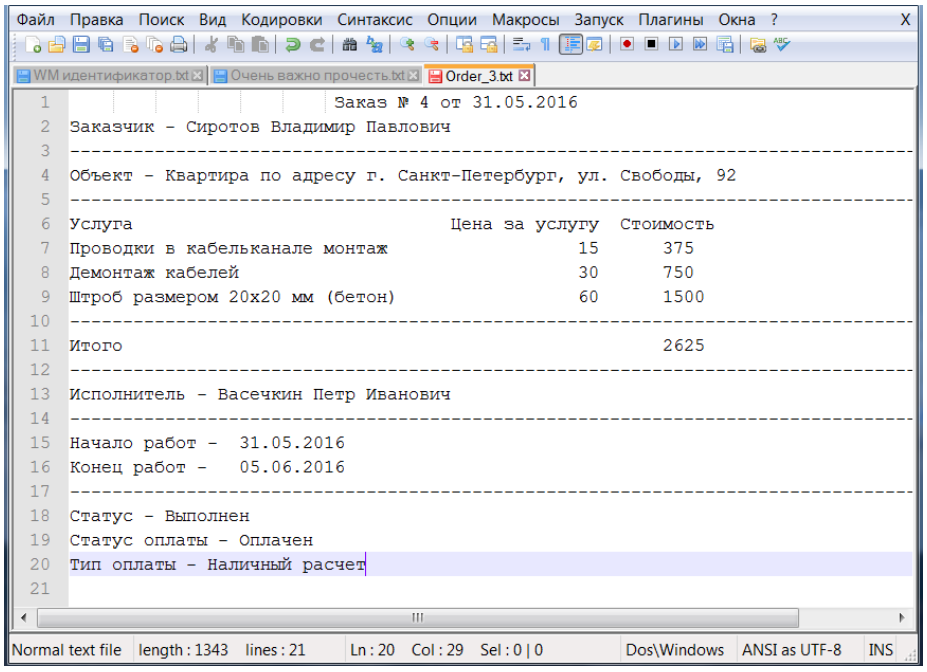


Рисунок 33 - Сохранение заказа в файл

Для того чтобы открыть меню поиска по заказам, выбираем «Поиск». Поиск возможен по трем критериям, это номер заказа, дата или по фамилии заказчика, (рис. 34).

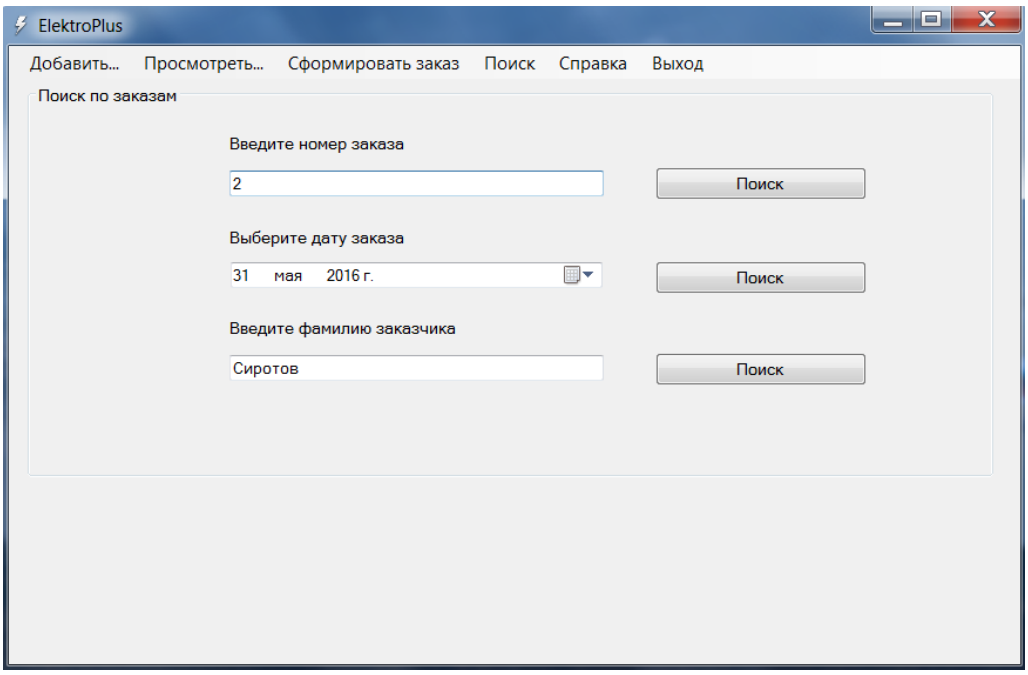


Рисунок 34 - Форма поиска заказов

В меню программы «Справка» предусмотрена инструкция для пользователей, в которой кратко описано, как работать с программой (рис. 35).

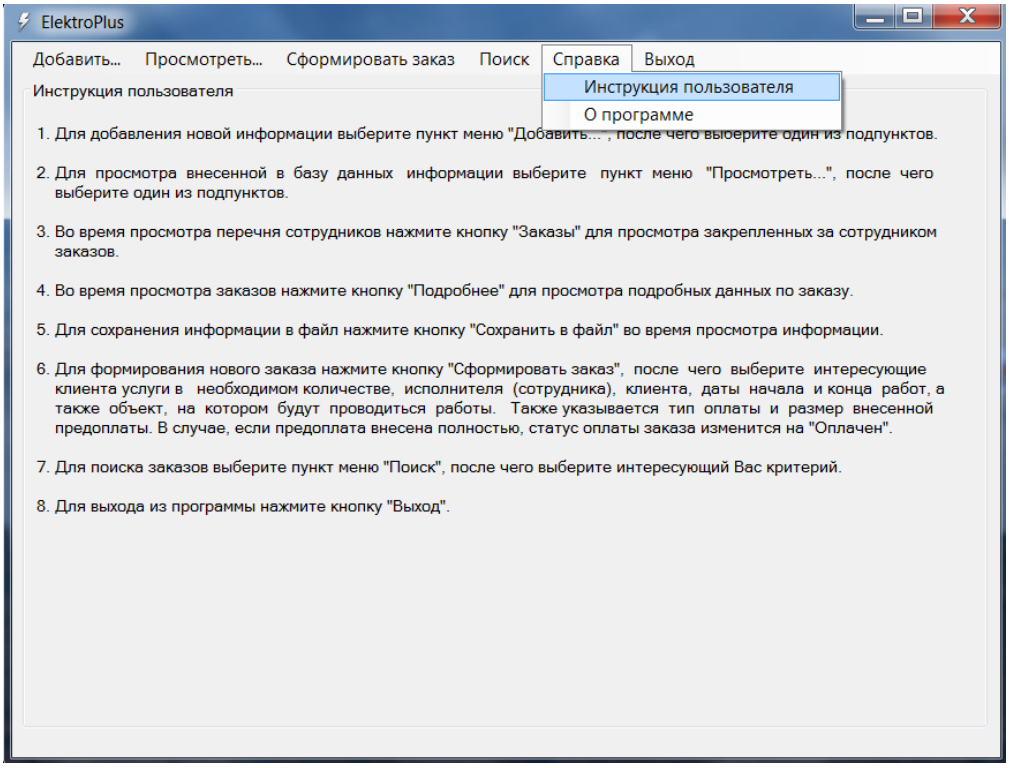


Рисунок 35 - Инструкция пользователя

4 Эргономика

Общей целью эргономики является обеспечение удобства и комфортных условий для эффективной деятельности человека (оператора), эффективное функционирование систем «человек — машина», а также обеспечение условий для сохранения здоровья и развитие личности человека (оператора).

Поэтому все большее количество разработчиков программных приложений (информационных систем) учитывают вопросы эргономики и юзабилити пользовательских интерфейсов и удобства работы. Юзабилити (применимость) информационных систем означает, что пользователи могут быстро и легко выполнять поставленные задачи, не обременяя себя долгим изучением пользовательского интерфейса[13].

Эргономичный пользовательский интерфейс должен удовлетворять следующим требованиям:

- способствовать быстрому освоению пользователем работы с программным приложением (информационной системой) и формировать у пользователя стандартные навыки работы;
- обеспечивать ввод информации пользователем наиболее удобным для него способом, не заботясь о ходе вычислений;
- обеспечивать согласование требований программного приложения (информационной системы), средств ввода и вывода информации с требованиями пользователя (информация должна быть понятной пользователю, объем представляемой информации должен быть согласован с объемом оперативной памяти пользователя);
- обеспечивать интуитивное и легкое управление программным приложением (информационной системой) пользователем;
- все время работы информационной системы пользовательский интерфейс должен находиться под контролем пользователя, при этом

никакие его действия не должны приводить к прерыванию работы программного приложения (информационной системы);

- обеспечивать исправление ошибок при вводе исходных данных без повторения ввода данных (в информации об ошибках следует делать акцент не на неправильные действия оператора, а на то, чем и каким образом можно исправить возникшие ошибки);
- обеспечивать обратную связь пользователя с программным приложением (справочная подсистема должна обеспечивать пользователя информацией, которая позволит настраивать работу с диалоговыми окнами, идентифицировать и устранять ошибки и определять порядок дальнейшей работы).

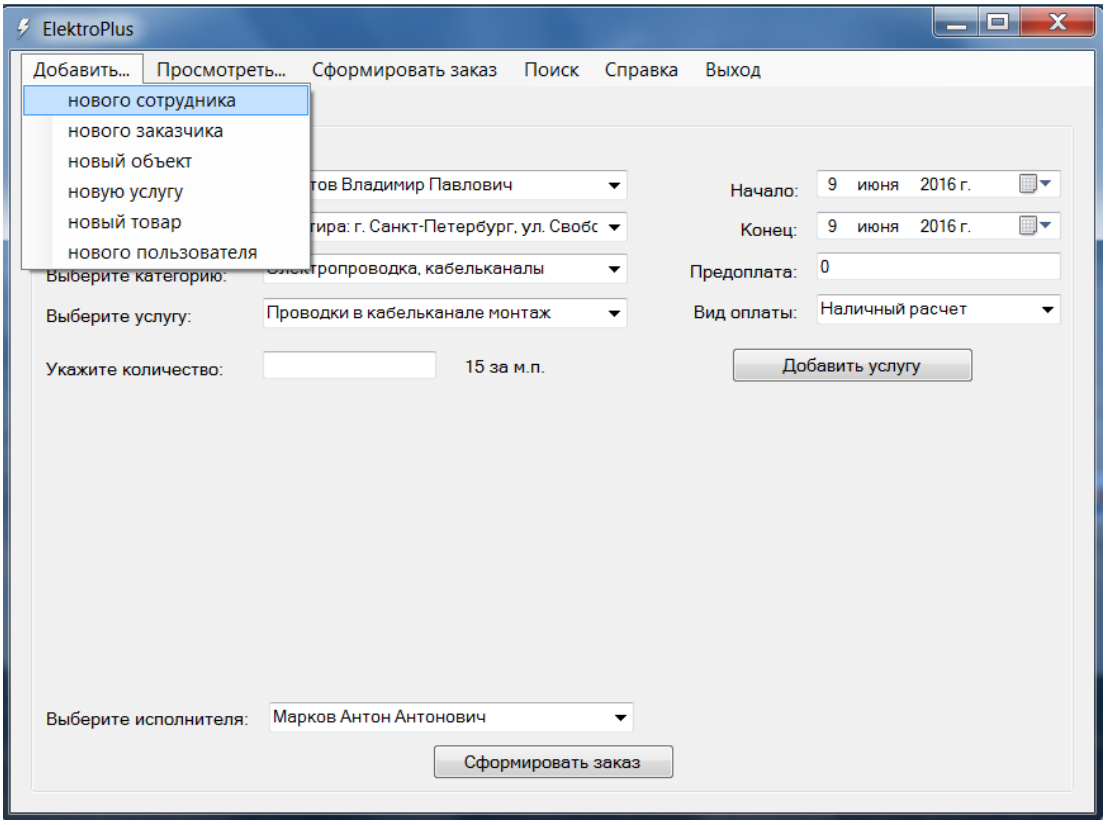


Рисунок 36 – Интерфейс информационной системы

Как видно на рисунке 36, автоматизированная информационная система обладает простым, интуитивным интерфейсом. Меню программы вполне понятное, в нем невозможно запутаться, кнопки находятся в удобных местах.

При ошибочных действиях пользователя, программа выдаст окно с сообщением об ошибке, чтобы предотвратить неверный ввод данных: (рис. 37).

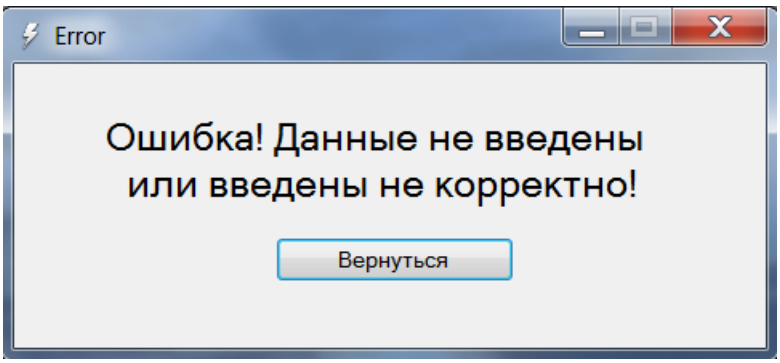


Рисунок 37 – Сообщение об ошибке

Для удобства взаимодействия с программой предусмотрена инструкция для пользователей, помогающая разобраться с программой (рис. 38).

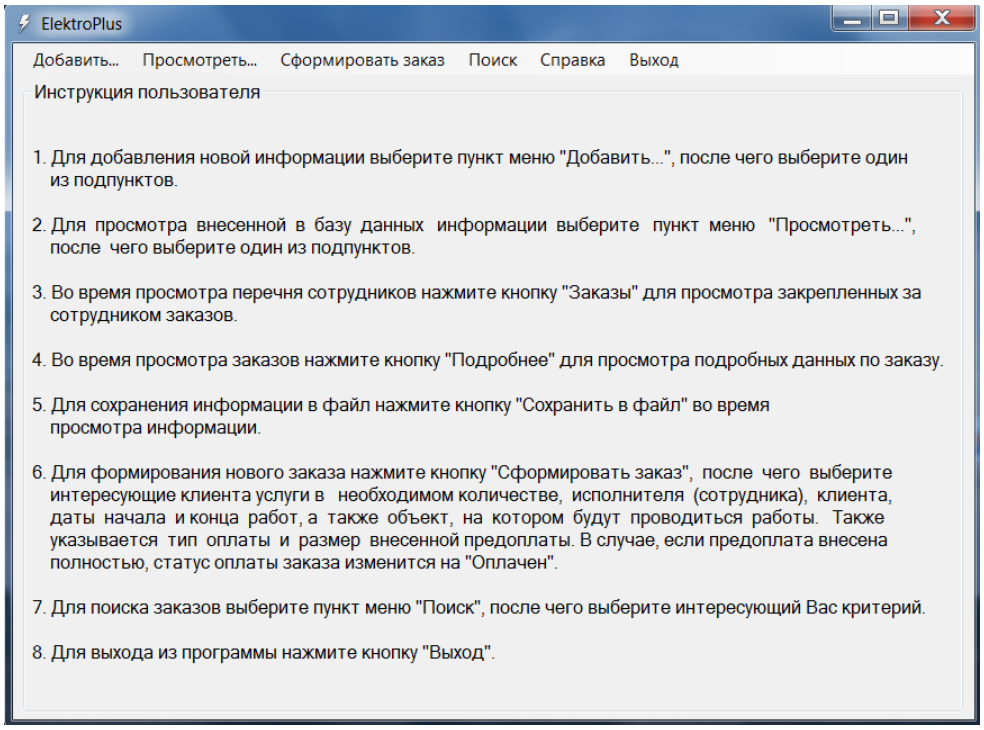


Рисунок 38 – Инструкция пользователя

В целом эргономика приложения соответствует стандартам и может считаться приемлемой, в процессе пользования программой пользователем.

ЗАКЛЮЧЕНИЕ

Разработанная система автоматизированного учета и контроля исполнения заказов существенно облегчает деятельность менеджеров, руководителей и персонала предприятия, позволяет клиентам получать своевременную информацию об услугах компании.

Использование системы автоматизированного учета и контроля исполнения заказов позволяет ускорить доступ к данным, сократить количество информации на бумажных носителях.

Дипломный проект был составлен и выполнен в согласии со стандартами, принятыми в учебном заведении [14].

В процессе выполнения дипломного проекта были применены уже имеющиеся, а также получены новые знания и умения для дальнейшей деятельности, практические навыки разработки баз данных и пользовательского интерфейса в среде разработки. Как результат, реализованная информационная система обладает удобным и понятным интерфейсом. Данная автоматизация позволит снизить издержки автоматизируемого предприятия.

В будущем информационную систему планируется усовершенствовать следующими технологиями:

- возможность работы в сети, одновременный доступ к одной базе с разных компьютеров;
- встроить функции создания резервной копии и восстановления базы данных;
- возможность сохранения бланков заказов в файлы Microsoft Office (Excel, Word).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Когаловский, М. Р. Перспективные технологии информационных систем. - М.: ДМК Пресс; Компания АйТи, 2003. — 288 с.

2 Петров, В. Н. Информационные системы - СПб.: Питер, 2002. – 688 с.

3 Никитина М.И. Системы и технологии поддержки принятия решений/ учебное пособие. Красноярск: ИПЦ КГТУ, 2005. 131 с.

4 С.Д. Кузнецов. Стандарты языка реляционных баз данных SQL: краткий обзор. / СУБД, 1996, N2, с. 6-36.

5 Дж. Мартин. Организация баз данных в вычислительных системах. - М.;Мир,1980. - 662с.

6 Уотсон, К. Visual C# 2010. Полный курс /Нейгел К., Педерсен Я.Х., Рид Дж., Скиннер М.- Вильямс, 2011г.- 955 с.

7 Гольцман,В. MySQL 5.0. Библиотека программиста – СПб: Питер, 2010 -253 с.

8 Справочное руководство по MySQL [Электронный ресурс] // Документация по MySQL. – Режим доступа: http://www.mysql.ru/docs/man/ (дата обращения: 02.06.2016 г.)

9 Зиборов, В.В. Visual C# 2012 на примерах - БХВ-Петербург, 2013г.- 475 с.

10 Либерти, Д. Программирование на C#. - СПб: Символ-Плюс, 2003г. – 688 с.

11 Дейтл, Х. C#: Пер. с англ. / Дейтел Х., Дейтел П., Листфилд Дж., Нието Т., Йегер Ш., Златкина М. — СПб.: БХВ-Петербург, 2006. — 1056 с.

12 Боуман Дж., Эмерсон С., Дарновски М. Практическое руководство по SQL, - Вильямс – 2002 г. - 322 с.

13 Попов А.А. Эргономика пользовательских интерфейсов в информационных системах: учебное пособие – РУСАЙНС, 2016.- 312 с.

14 СТО 4.2–07–2014 Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности. – Введ. 09.01.2014. – Красноярск: ИПК СФУ, 2014. – 60 с.

ПРИЛОЖЕНИЕ А

Листинги программных модулей

Листинг класса Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ElektroPlus
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Autorization());
        }
    }
}
```

ПРИЛОЖЕНИЕ Б

Листинг модуля MainForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace ElektroPlus
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
            new DataBaseConnect();
        }
        List<Personal> personal = new SelectToDataBase().getPersonal();
        List<Service> service = new SelectToDataBase().getService();
```

```
List<Customer> customer = new SelectToDataBase().getCustomer();
List<WObject> wObject = new SelectToDataBase().getWObject();
Order order = new Order();
List<Order> orders = new List<Order>();
int cost;


//Метод для сериализации заказа
public static byte[] SaveOrder(object objGraph)
{
    MemoryStream binF = new MemoryStream();
    BinaryFormatter binFormat = new BinaryFormatter();
    using (binF)
    {
        binFormat.Serialize(binF, objGraph);
    }
    return binF.ToArray();
}


//Метод для десериализации заказа
public void LoadOrder(List<byte[]> binaryData)
{
    orders.Clear();
    for (int i = 0; i < binaryData.Count; i++)
    {
        BinaryFormatter formatter = new BinaryFormatter();
        MemoryStream ms = new MemoryStream(binaryData[i]);
        orders.Add((Order)formatter.Deserialize(ms));
    }
    return;
}

private void новогоСотрудникаToolStripMenuItem_Click(object sender, EventArgs e)
{
    groupBox1.Visible = true;
    groupBox2.Visible = false;
    groupBox3.Visible = false;
    groupBox4.Visible = false;
    groupBox5.Visible = false;
    groupBox6.Visible = false;
    groupBox7.Visible = false;
    groupBox8.Visible = false;
    groupBox9.Visible = false;
    groupBox10.Visible = false;
    groupBox11.Visible = false;
    groupBox12.Visible = false;
    groupBox13.Visible = false;
    groupBox14.Visible = false;
    dataGridView1.Visible = false;
    dataGridView2.Visible = false;
    dataGridView3.Visible = false;
    dataGridView4.Visible = false;
    dataGridView5.Visible = false;

}

private void button1_Click(object sender, EventArgs e)
{
```

```
            if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" ||
textBox4.Text == "" || textBox5.Text == "" || textBox6.Text == "" || textBox7.Text == ""
|| textBox8.Text == "" || textBox9.Text == "" || textBox10.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {
                new InsertToDataBase().addPersonal(textBox1.Text, textBox2.Text,
textBox3.Text, textBox4.Text, textBox5.Text, textBox6.Text, textBox7.Text, textBox8.Text,
textBox9.Text, textBox10.Text);
                textBox1.Text = "";
                textBox2.Text = "";
                textBox3.Text = "";
                textBox4.Text = "";
                textBox5.Text = "";
                textBox6.Text = "";
                textBox7.Text = "";
                textBox8.Text = "";
                textBox9.Text = "";
                textBox10.Text = "";
                personal.Clear();
                personal = new SelectToDataBase().getPersonal();
            }
        }

        private void новогоЗаказчикаToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = true;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (textBox11.Text == "" || textBox12.Text == "" || textBox13.Text == "" ||
textBox14.Text == "" || textBox15.Text == "" || textBox16.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
```

```
            {

                new InsertToDataBase().addCustomer(textBox11.Text, textBox12.Text,
textBox13.Text, textBox14.Text, textBox15.Text, textBox16.Text);
                textBox11.Text = "";
                textBox12.Text = "";
                textBox13.Text = "";
                textBox14.Text = "";
                textBox15.Text = "";
                textBox16.Text = "";
                customer.Clear();
                customer = new SelectToDataBase().getCustomer();
            }
        }

        private void новыйОбъектToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = true;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (textBox17.Text == "" || textBox18.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {

                new InsertToDataBase().addObject(textBox17.Text, textBox18.Text);
                textBox17.Text = "";
                textBox18.Text = "";
                wObject.Clear();
                wObject = new SelectToDataBase().getWObject();
            }
        }

        private void новыйToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
```

```
            groupBox3.Visible = false;
            groupBox4.Visible = true;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            if (textBox19.Text == "" || textBox20.Text == "" || textBox21.Text == "" ||
textBox22.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {

                new InsertToDataBase().addService(textBox19.Text, textBox20.Text,
textBox21.Text, textBox22.Text);
                textBox19.Text = "";
                textBox20.Text = "";
                textBox21.Text = "";
                textBox22.Text = "";
                service.Clear();
                service = new SelectToDataBase().getService();
            }
        }

        private void новыйТоварToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = true;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
```

```
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;

        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (textBox23.Text == "" || textBox24.Text == "" || textBox25.Text == "" ||
textBox26.Text == "" || textBox27.Text == "" || textBox28.Text == "" || textBox29.Text ==
"")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {

                new InsertToDataBase().addTovar(textBox23.Text, textBox24.Text,
textBox25.Text, textBox26.Text, textBox27.Text, textBox28.Text, textBox29.Text);
                textBox23.Text = "";
                textBox24.Text = "";
                textBox25.Text = "";
                textBox26.Text = "";
                textBox27.Text = "";
                textBox28.Text = "";
                textBox29.Text = "";
            }
        }

        private void сформироватьЗаказToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = true;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            comboBox1.Items.Clear();

            for (int i = 0; i < customer.Count; i++)
            {
                comboBox1.Items.Add(customer[i].surname_c + " " + customer[i].name_c + " "
+ customer[i].sname_c);//добавить строку в позицию
            }
            comboBox1.SelectedIndex = 0;
            comboBox4.Items.Clear();
            for (int i = 0; i < personal.Count; i++)
```

```
            {
                comboBox4.Items.Add(personal[i].surname_p + " " + personal[i].name_p + " "
+ personal[i].sname_p);//добавить строку в позицию
            }
            comboBox4.SelectedIndex = 0;
            comboBox2.Items.Clear();
            for (int i = 0; i < service.Count; i++)
            {
                if (!comboBox2.Items.Contains(service[i].category_s))
                {
                    comboBox2.Items.Add(service[i].category_s);
                }
            }
            comboBox2.SelectedIndex = 0;
            comboBox3.Items.Clear();
            for (int i = 0; i < service.Count; i++)
            {
                if (comboBox2.SelectedItem.Equals(service[i].category_s))
                {
                    comboBox3.Items.Add(service[i].name_s);
                }
            }
            comboBox3.SelectedIndex = 0;
            for (int i = 0; i < service.Count; i++)
            {
                if (comboBox3.SelectedItem.Equals(service[i].name_s))
                {
                    label36.Text = service[i].cost_s + " за " + service[i].unit_s;
                    cost = service[i].cost_s;
                    break;
                }
            }
            comboBox5.Items.Clear();
            for (int i = 0; i < wObject.Count; i++)
            {
                comboBox5.Items.Add(wObject[i].name_o + ": " +
wObject[i].adres_o);//добавить строку в позицию
            }
            comboBox5.SelectedIndex = 0;
            comboBox7.Items.Clear();
            comboBox7.Items.Add("Наличный расчет");
            comboBox7.Items.Add("Безналичный расчет");
            comboBox7.SelectedIndex = 0;
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            comboBox3.Items.Clear();
            for (int i = 0; i < service.Count; i++)
            {
                if (comboBox2.SelectedItem.Equals(service[i].category_s))
                {
                    comboBox3.Items.Add(service[i].name_s);
                }
            }
            comboBox3.SelectedIndex = 0;
            for (int i = 0; i < service.Count; i++)
            {
                if (comboBox3.SelectedItem.Equals(service[i].name_s))
                {
                    label36.Text = service[i].cost_s + " руб. за " + service[i].unit_s;
```

```
                cost = service[i].cost_s;
                break;
            }
        }
    }

    private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
    {
        for (int i = 0; i < service.Count; i++)
        {
            if (comboBox3.SelectedItem.Equals(service[i].name_s))
            {
                label36.Text = service[i].cost_s + " за " + service[i].unit_s;
                cost = service[i].cost_s;
                break;
            }
        }
    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (textBox31.Text == "")
        {
            int index = 1;
            Error usr = new Error(index);
            usr.Visible = true;
        }
        else
        {
            order.addService(Convert.ToString(comboBox3.SelectedItem),
Convert.ToInt32(textBox31.Text), cost);
            dataGridView1.Visible = true;
            dataGridView1.Rows.Clear();
            for (int i = 0; i < order.service.Count; i++)
            {
                dataGridView1.Rows.Add(order.service[i].name_s,
order.service[i].category_s, order.cost[i]);
            }
            label48.Text = "ИТОГО: " + order.total + " руб.";
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        if (order.service.Count == 0)
        {
            int index = 2;
            Error usr = new Error(index);
            usr.Visible = true;
        }
        else
        {
            order.addCustomer(Convert.ToString(comboBox1.SelectedItem));
            order.addObject(Convert.ToString(comboBox5.SelectedItem));
            order.addPersonal(Convert.ToString(comboBox4.SelectedItem));
            order.index = new SelectToDataBase().getLastIndex();
            order.cashType = (Convert.ToString(comboBox7.SelectedItem));
            order.date = DateTime.Today;
            order.sDate = Convert.ToDateTime(dateTimePicker1.Text);
            order.fDate = Convert.ToDateTime(dateTimePicker2.Text);
            if (Convert.ToInt32(textBox30.Text) > order.total)
```

```
                {
                    int index = 1;
                    Error usr = new Error(index);
                    usr.Visible = true;
                }
                else if (Convert.ToInt32(textBox30.Text) == order.total)
                {
                    order.cashStatus = "Оплачен";
                }
                else
                {
                    order.predoplata = Convert.ToInt32(textBox30.Text);
                }
                new InsertToDataBase().addOrder(SaveOrder(order));
                textBox1.Text = "";
                dataGridView1.Visible = false;
                dataGridView1.Rows.Clear();
                order = new Order();
                сформироватьЗаказToolStripMenuItem_Click(new object(), new EventArgs());
            }
        }

        private void переченьСотрудниковToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = true;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = true;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            dataGridView2.Rows.Clear();
            for (int i = 0; i < personal.Count; i++)
            {
                dataGridView2.Rows.Add(personal[i].surname_p + " " + personal[i].name_p +
" " + personal[i].sname_p, personal[i].doljnost_p, personal[i].bdate_p,
personal[i].range_p + " " + personal[i].number_p, personal[i].telephone_p);
            }
        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
            Application.Exit();
        }

        private void переченьОбъектовToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = true;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = true;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            dataGridView3.Rows.Clear();
            for (int i = 0; i < wObject.Count; i++)
            {
                dataGridView3.Rows.Add(wObject[i].name_o, wObject[i].adres_o);
            }
        }

        private void перечеңЗаказчиковToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = true;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = true;
            dataGridView5.Visible = false;
            dataGridView4.Rows.Clear();
            for (int i = 0; i < customer.Count; i++)
            {
                dataGridView4.Rows.Add(customer[i].surname_c + " " + customer[i].name_c +
" " + customer[i].sname_c, customer[i].telephone_c, customer[i].email_c,
customer[i].adres_c);
            }
        }

        private void новогоПользователяToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
```

```
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = true;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            comboBox6.Items.Clear();
            comboBox6.Items.Add("Сотрудник");
            comboBox6.Items.Add("Клиент");
            comboBox6.SelectedIndex = 0;
        }

        private void button8_Click(object sender, EventArgs e)
        {
            if (textBox33.Text == "" || textBox34.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {
                new InsertToDataBase().addUser(textBox33.Text, textBox34.Text,
Convert.ToString(comboBox6.SelectedItem));
                textBox33.Text = "";
                textBox34.Text = "";
            }
        }

        private void переченьЗаказовToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = true;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = true;
            LoadOrder(new SelectToDataBase().getOrder());
```

```
            dataGridView5.Rows.Clear();
            for (int i = 0; i < orders.Count; i++)
            {
                dataGridView5.Rows.Add(orders[i].index, orders[i].customer.surname_c + " "
+ orders[i].customer.name_c + " " + orders[i].customer.sname_c,
orders[i].personal.surname_p + " " + orders[i].personal.name_p + " " +
orders[i].personal.sname_p, orders[i].status);
            }
        }

        public void dataGridView5_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            Order selectedOrder = new Order();
            for (int i = 0; i < orders.Count; i++)
            {
                if (orders[i].index ==
(int)dataGridView5.Rows[dataGridView5.CurrentCell.RowIndex].Cells[0].Value)
                {
                    selectedOrder = orders[i];
                    break;
                }
            }
            OrderForm usr = new OrderForm(selectedOrder);
            usr.Visible = true;

        }

        private void переченьТоваровToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ProductForm usr = new ProductForm();
            usr.Visible = true;
        }

        private void поискToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = true;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
            textBox36.Text = "";
            textBox32.Text = "";
        }

        private void button9_Click(object sender, EventArgs e)
        {
```

```
            if (textBox32.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {
                groupBox1.Visible = false;
                groupBox2.Visible = false;
                groupBox3.Visible = false;
                groupBox4.Visible = false;
                groupBox5.Visible = false;
                groupBox6.Visible = false;
                groupBox7.Visible = false;
                groupBox8.Visible = false;
                groupBox9.Visible = false;
                groupBox10.Visible = false;
                groupBox11.Visible = true;
                groupBox12.Visible = false;
                groupBox13.Visible = false;
                groupBox14.Visible = false;
                dataGridView1.Visible = false;
                dataGridView2.Visible = false;
                dataGridView3.Visible = false;
                dataGridView4.Visible = false;
                dataGridView5.Visible = true;
                LoadOrder(new SelectToDataBase().getOrder());
                dataGridView5.Rows.Clear();
                for (int i = 0; i < orders.Count; i++)
                {
                    if (orders[i].index == Convert.ToInt32(textBox32.Text))
                    {
                        dataGridView5.Rows.Add(orders[i].index,
orders[i].customer.surname_c + " " + orders[i].customer.name_c + " " +
orders[i].customer.sname_c, orders[i].personal.surname_p + " " + orders[i].personal.name_p
+ " " + orders[i].personal.sname_p, orders[i].status);
                    }
                }
            }
        }

        private void button11_Click(object sender, EventArgs e)
        {
            if (textBox36.Text == "")
            {
                int index = 1;
                Error usr = new Error(index);
                usr.Visible = true;
            }
            else
            {
                groupBox1.Visible = false;
                groupBox2.Visible = false;
                groupBox3.Visible = false;
                groupBox4.Visible = false;
                groupBox5.Visible = false;
                groupBox6.Visible = false;
                groupBox7.Visible = false;
                groupBox8.Visible = false;
                groupBox9.Visible = false;
```

```
                groupBox10.Visible = false;
                groupBox11.Visible = true;
                groupBox12.Visible = false;
                groupBox13.Visible = false;
                groupBox14.Visible = false;
                dataGridView1.Visible = false;
                dataGridView2.Visible = false;
                dataGridView3.Visible = false;
                dataGridView4.Visible = false;
                dataGridView5.Visible = true;
                LoadOrder(new SelectToDataBase().getOrder());
                dataGridView5.Rows.Clear();
                for (int i = 0; i < orders.Count; i++)
                {
                    if
(orders[i].customer.surname_c.Contains(Convert.ToString(textBox36.Text)))
                    {
                        dataGridView5.Rows.Add(orders[i].index,
orders[i].customer.surname_c + " " + orders[i].customer.name_c + " " +
orders[i].customer.sname_c, orders[i].personal.surname_p + " " + orders[i].personal.name_p
+ " " + orders[i].personal.sname_p, orders[i].status);
                    }
                }
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = true;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = true;
            LoadOrder(new SelectToDataBase().getOrder());
            dataGridView5.Rows.Clear();
            for (int i = 0; i < orders.Count; i++)
            {
                if (orders[i].date == Convert.ToDateTime(dateTimePicker3.Text))
                {
                    dataGridView5.Rows.Add(orders[i].index, orders[i].customer.surname_c +
" " + orders[i].customer.name_c + " " + orders[i].customer.sname_c,
orders[i].personal.surname_p + " " + orders[i].personal.name_p + " " +
orders[i].personal.sname_p, orders[i].status);
                }
            }
        }
```

```
        private void button12_Click(object sender, EventArgs e)
        {
            string t = "";
            string p = "Files/Customers.txt";
            using (StreamWriter output = File.CreateText(p))
            {
                output.WriteLine("\t\t\t\t    Клиенты ООО \"ЭлектроПлюс\"");
                output.WriteLine();
                output.WriteLine();
                for (int j = 0; j < customer.Count; j++)
                {
                    output.WriteLine("-------------------------------------------------
-----------------------------------");
                    if ((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 40)
                        t = "\t";
                    else if (((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 32) & ((customer[j].surname_c + " " + customer[j].name_c +
" " + customer[j].sname_c).Length < 40))
                        t = "\t\t";
                    else if (((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 24) & ((customer[j].surname_c + " " + customer[j].name_c +
" " + customer[j].sname_c).Length < 32))
                        t = "\t\t\t";
                    else if (((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 16) & ((customer[j].surname_c + " " + customer[j].name_c +
" " + customer[j].sname_c).Length < 24))
                        t = "\t\t\t\t";
                    else if (((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 8) & ((customer[j].surname_c + " " + customer[j].name_c + "
" + customer[j].sname_c).Length < 16))
                        t = "\t\t\t\t\t";
                    else if (((customer[j].surname_c + " " + customer[j].name_c + " " +
customer[j].sname_c).Length >= 0) & ((customer[j].surname_c + " " + customer[j].name_c + "
" + customer[j].sname_c).Length < 8))
                        t = "\t\t\t\t\t\t";
                    output.WriteLine("" + customer[j].surname_c + " " + customer[j].name_c
+ " " + customer[j].sname_c + t + customer[j].telephone_c + "\t" + customer[j].email_c);
                    output.WriteLine("" + customer[j].adres_c);
                    output.WriteLine("-------------------------------------------------
-----------------------------------");
                }

                output.Close();
            }
            System.Diagnostics.Process.Start(".\\Files\\Customers.txt");
        }

        private void button13_Click(object sender, EventArgs e)
        {
            string t = "";
            string p = "Files/Personal.txt";
            using (StreamWriter output = File.CreateText(p))
            {
                output.WriteLine("\t\t\t    Сотрудники ООО \"ЭлектроПлюс\"");
                output.WriteLine();
                output.WriteLine();
                for (int j = 0; j < personal.Count; j++)
                {
                    output.WriteLine("-------------------------------------------------
----------------------------");
```

```
                    if ((personal[j].surname_p + " " + personal[j].name_p + " " +
personal[j].sname_p).Length >= 32)
                        t = "\t";
                    else if (((personal[j].surname_p + " " + personal[j].name_p + " " +
personal[j].sname_p).Length >= 24) & ((personal[j].surname_p + " " + personal[j].name_p +
" " + personal[j].sname_p).Length < 32))
                        t = "\t\t";
                    else if (((personal[j].surname_p + " " + personal[j].name_p + " " +
personal[j].sname_p).Length >= 16) & ((personal[j].surname_p + " " + personal[j].name_p +
" " + personal[j].sname_p).Length < 24))
                        t = "\t\t\t";
                    else if (((personal[j].surname_p + " " + personal[j].name_p + " " +
personal[j].sname_p).Length >= 8) & ((personal[j].surname_p + " " + personal[j].name_p + "
" + personal[j].sname_p).Length < 16))
                        t = "\t\t\t\t";
                    else if (((personal[j].surname_p + " " + personal[j].name_p + " " +
personal[j].sname_p).Length >= 0) & ((personal[j].surname_p + " " + personal[j].name_p + "
" + personal[j].sname_p).Length < 8))
                        t = "\t\t\t\t\t";
                    output.WriteLine("" + personal[j].doljnost_p);
                    output.WriteLine("" + personal[j].surname_p + " " + personal[j].name_p
+ " " + personal[j].sname_p + t + personal[j].bdate_p + "\t" + personal[j].telephone_p +
"\t" + personal[j].range_p + personal[j].number_p);
                    output.WriteLine("" + personal[j].adres_p);
                    output.WriteLine("----------------------------------------------------
----------------------------");
                }

                output.Close();
            }
            System.Diagnostics.Process.Start(".\\Files\\Personal.txt");
        }

        private void button14_Click(object sender, EventArgs e)
        {
            string t = "";
            string p = "Files/Objects.txt";
            using (StreamWriter output = File.CreateText(p))
            {
                output.WriteLine("\t\t\t    Объекты ООО \"ЭлектроПлюс\"");
                output.WriteLine();
                output.WriteLine();
                for (int j = 0; j < wObject.Count; j++)
                {
                    output.WriteLine("----------------------------------------------------
----------------------------");
                    if (wObject[j].name_o.Length >= 32)
                        t = "\t";
                    else if ((wObject[j].name_o.Length >= 24) & (wObject[j].name_o.Length
< 32))
                        t = "\t\t";
                    else if ((wObject[j].name_o.Length >= 16) & (wObject[j].name_o.Length
< 24))
                        t = "\t\t\t";
                    else if ((wObject[j].name_o.Length >= 8) & (wObject[j].name_o.Length <
16))
                        t = "\t\t\t\t";
                    else if ((wObject[j].name_o.Length >= 0) & (wObject[j].name_o.Length <
8))
                        t = "\t\t\t\t\t";
                    output.WriteLine("" + wObject[j].name_o + t + wObject[j].adres_o);
```

```
                    output.WriteLine("----------------------------------------------------
-----------------------------");
                }

                output.Close();
            }
            System.Diagnostics.Process.Start(".\\Files\\Objects.txt");
        }

        private void dataGridView2_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = true;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = true;
            LoadOrder(new SelectToDataBase().getOrder());
            dataGridView5.Rows.Clear();
            for (int i = 0; i < orders.Count; i++)
            {
                if ((orders[i].personal.surname_p + " " + orders[i].personal.name_p + " "
+ orders[i].personal.sname_p).Equals
((string)dataGridView2.Rows[dataGridView2.CurrentCell.RowIndex].Cells[0].Value))
                    dataGridView5.Rows.Add(orders[i].index, orders[i].customer.surname_c +
" " + orders[i].customer.name_c + " " + orders[i].customer.sname_c,
orders[i].personal.surname_p + " " + orders[i].personal.name_p + " " +
orders[i].personal.sname_p, orders[i].status);
            }
        }

        private void инструкцияПользователяToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = true;
```

```
            groupBox14.Visible = false;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
        }

        private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            groupBox1.Visible = false;
            groupBox2.Visible = false;
            groupBox3.Visible = false;
            groupBox4.Visible = false;
            groupBox5.Visible = false;
            groupBox6.Visible = false;
            groupBox7.Visible = false;
            groupBox8.Visible = false;
            groupBox9.Visible = false;
            groupBox10.Visible = false;
            groupBox11.Visible = false;
            groupBox12.Visible = false;
            groupBox13.Visible = false;
            groupBox14.Visible = true;
            dataGridView1.Visible = false;
            dataGridView2.Visible = false;
            dataGridView3.Visible = false;
            dataGridView4.Visible = false;
            dataGridView5.Visible = false;
        }
    }
}
```